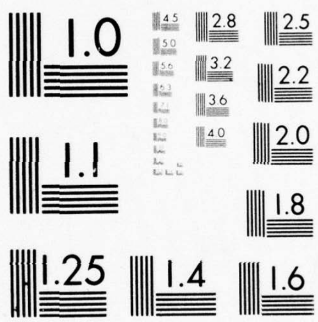
AD-A046 598 COLORADO STATE UNIV FORT COLLINS DEPT OF MATHEMATICS F/G 12/1
ADAPTIVE CURVE FITTING.(U)
JUL 77 J A HULL, G D TAYLOR AFOSR-76-2878
UNCLASSIFIED AFOSR-TR-77-1257 NL

| OF |
AD
A046598

END
DATE
FILMED
12-77
DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

AD A046598

DDC FILE COPY

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFOSR-TR- 77-1257	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): ADAPTIVE CURVE FITTING	5. TYPE OF REPORT & PERIOD COVERED: Interim rept.
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): J. A. Hull ~~and~~ G. D. Taylor	8. CONTRACT OR GRANT NUMBER(s): AFOSR-76-2878
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Colorado State University, Department of Mathematics, Fort Collins, CO 80523	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F, 2304/A3
11. CONTROLLING OFFICE NAME AND ADDRESS: AFOSR/NM, Bldg. 410, Bolling AFB, D.C. 20332	12. REPORT DATE: July 1977
	13. NUMBER OF PAGES: 45
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): 47p.	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

D D C
RECEIVED
NOV 21 1977
F

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Curve fitting, data fitting

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper ~~we~~ present an algorithm for adaptively computing smooth piecewise polynomial approximations which uses either the best uniform or best (discrete) $L^{(2)}$ approximation operator as its local approximation operator. ~~We do not require any~~ No knowledge of derivatives of the function being approximated is required. Due to the approximation properties of the respective operators, ~~we have found that~~ the algorithm found using best uniform approximations is particularly suited for approximating precise mathematical functions and the algorithm using best $L^{(2)}$

→ next page

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

407344

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

20 approximations is particularly suited for approximating data with significant levels of noise. Finally, this algorithm can be used with classes of approximating functions other than polynomials. Fortran codes for these two algorithms are given in the appendix at the end of this paper.

White Section
Buff Section
DISTRIBUTION/AVAILABILITY CODES
SPECIAL
A 23 E.G.

AFOSR-TR- 77-1257

Approved for public release;
distribution unlimited.

ADAPTIVE CURVE FITTING

by

J. A. Hull[1] and G. D. Taylor[2]

ABSTRACT

In this paper we present an algorithm for adaptively computing smooth piecewise polynomial approximations which uses either the best uniform or best (discrete) L^2 approximation operator as its local approximation operator. We do not require any knowledge of derivatives of the function being approximated. Due to the approximation properties of the respective operators, we have found that the algorithm using best uniform approximations is particularly suited for approximating precise mathematical functions and the algorithm using best L^2 approximations is particularly suited for approximating data with significant levels of noise. Finally, this algorithm can be used with classes of approximating functions other than polynomials.

Send proofs to G. D. Taylor

[1] Applied Technology
645 Almanor
Sunnyvale, California 94086

[2] Department of Mathematics
Colorado State University
Fort Collins, Colorado 80523

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH (AFSC)
NOTICE OF TRANSMITTAL TO DDC
This technical report has been reviewed and is approved for public release IAW AFR 190-12 (7b).
Distribution is unlimited.
A. D. BLOSE
Technical Information Officer

Research sponsored by the Air Force Office of Scientific Research, Air Force Systems, USAF, under grant number 76-2878.

CR category numbers: 5.13.

Key words and phrases: curve fitting, data fitting, approximation.

I. Introduction

Let X be a finite set of real points and let f be a function defined on X (given in tabular form) which we desire to approximate. Let $a = \min\{x: x \in X\}$ and $b = \max\{x: x \in X\}$ and for any function g defined on X define $\|g\|_X = \max\{|g(x)|: x \in X\}$. Finally, let TOL, SMTH and N be parameters supplied by the user where TOL is a positive number, SMTH is a nonnegative integer and N-1 is an integer greater than or equal to SMTH. In this setting, our algorithm will calculate an approximation, p, to f and a set of points $\{x_i\}_{i=0}^{k} \subset X$ with $a = x_0 < x_1 < \dots < x_k = b$ such that p restricted to $[x_i, x_{i+1}]$ is a polynomial $p_i \in \Pi_{N-1} = \{q: q$ is a real algebraic polynomial of degree $\leq N-1\}$, p has SMTH continuous derivatives and $\|f-p\|_X \leq$ TOL. In what follows, we shall use the notation $\|g\|_S$ to denote $\max\{|g(t)|: t \in S\}$ for all g defined on S and $S \subset X$.

II. The Algorithm

The algorithm begins by choosing $\tilde{x}_1$ to be the largest point in X such that

(1) $[a, \tilde{x}_1] \cap X$ contains at least max(2, N + 1) points, and

(2) If p_1 is the best (uniform or L^2) approximation to f from Π_{N-1} on $S_1 = [a, \tilde{x}_1] \cap X$ then $\|f - p_1\|_{S_1} \leq$ TOL.

If $\tilde{x}_1 = b$, then since p_1 is a piecewise polynomial meeting our requirements, we successfully terminate the algorithm. If no such $\tilde{x}_1$ exists, the algorithm fails and an appropriate error message is generated. Otherwise, if SMTH = 0 (i.e., we only require the approximation to be continuous) we choose the right endpoint of the first subinterval, x_1, to be $\tilde{x}_1$. If SMTH > 0, in order to add to the stability of the algorithm we in general choose x_1 by "backing off" from $\tilde{x}_1$ in the following manner.

We first examine the error curve $f(x) - p_1(x)$ and find those points $\xi_1, \xi_2, \dots, \xi_\ell$ in $(a, \tilde{x}_1] \cap X$ at which relative extrema occur. (Note that when

using a Remes-like algorithm to compute best uniform approximations with interpolatory constraints, these points are readily available). We will choose one of the ξ_ν's to be x_1. The motivation for choosing x_1 in this manner is that in the continuous setting, if f is differentiable and ξ is an (interior) relative extreme point of $f(x) - p_1(x)$ then $f'(\xi) - p_1'(\xi) = 0$ so that the derivative of p_1 at ξ would match that of f at ξ. This guarantees that when we smoothly join the next piece of the approximation to p_1 at ξ, this next piece will closely follow the direction of f at least near ξ. If we merely joined the second piece to the first at $\tilde{x}_1$ no such guarantee can be made and, in fact, severe oscillatory problems tend to set in. Our numerical experience indicates that the procedure of backing off from $\tilde{x}_1$ to a smaller x_1 contributes very significantly toward the stability of the algorithm. To continue, let $\tilde{f}'(\xi_\nu)$ be the derivative of the centered quadratic interpolation of f evaluated at ξ_ν. We then choose x_1 to be the largest ξ_ν such that $|\tilde{f}'(\xi_\nu) - p_1'(\xi_\nu)| < \text{EPS}$, where EPS is a user-definable prescribed tolerance, or, if there does not exist such a ξ_ν, then we let x_1 be the largest ξ_ν at which $|\tilde{f}'(\xi_\nu) - p_1'(\xi_\nu)|$ is a minimum. (In our implementation of the algorithm, we do not in general consider all of the relative extreme points of $f(x) - p_1(x)$ in $(a, \tilde{x}_1] \cap X$, but only the largest N-SMTH-1 of them.)

We continue the algorithm by finding successive intervals $[x_1, x_2]$, $[x_2, x_3]$, $\ldots$, $[x_{m-1}, b]$, and corresponding polynomial approximations $p_2, p_3, \ldots, p_m \in \Pi_{N-1}$ to f so that $p_{\nu-1}^{(j)}(x_{\nu-1}) = p_\nu^{(j)}(x_{\nu-1})$ for $j = 0, 1, \ldots$, SMTH and $\|f - p_\nu\|_{S_\nu} \le \text{TOL}$, where $S_\nu = [x_{\nu-1}, x_\nu] \cap X$ for $\nu = 2, \ldots, m$, $x_m = b$. This is accomplished as follows:

Suppose we have found the subintervals $[a, x_1]$, $[x_1, x_2]$, $\ldots$, $[x_{i-2}, x_{i-1}]$, and the corresponding approximations $p_1, p_2, \ldots, p_{i-1}$. Assume further that $[x_{i-1}, b] \subset X$ contains at least max(2, N-SMTH) points. We now determine an x_i and a p_i meeting the above requirements. We begin by choosing $\tilde{x}_i \in X$ to be the largest point in X which satisfies

(1) $[x_{i-1}, \tilde{x}_i] \cap X$ contains at least max(2, N-SMTH) points,

(2) If p_i is the appropriate best approximation to f from Π_{N-1} on $S_i = [x_{i-1}, \tilde{x}_i] \cap X$ subject to the constraint that $p_i^{(j)}(x_{i-1}) = p_{i-1}^{(j)}(x_{i-1})$, $j = 0, 1, \ldots,$ SMTH, then $\|f - p_i\|_{S_i} \leq$ TOL.

If $\tilde{x}_i = b$, we set $x_i = \tilde{x}_i = b$, and the algorithm is successfully terminated. If no such $\tilde{x}_i$ exists, the algorithm fails and is terminated. Otherwise, we choose x_i completely analogous to our choice of x_1 above.

Finally, we consider the special case where $[x_{i-1}, b] \cap X$ contains fewer than max(2, N-SMTH) points. In this case we replace x_{i-1} with some $\hat{x}_{i-1} \in X$, where $x_{i-2} < \hat{x}_{i-1} < x_{i-1} < b$, so that $[\hat{x}_{i-1}, b] \cap X$ contains at least max(2, N-SMTH) points. Specifically, we choose $\hat{x}_{i-1}$ to be a point in X closest to $(b - x_{i-2})/2$ which satisfies

(1) $[\hat{x}_{i-1}, b] \cap X$ contains at least max(2, N-SMTH) points.

(2) If p_i is the appropriate best approximation to f from Π_{N-1} on $\hat{S} = [\hat{x}_{i-1}, b] \cap X$ subject to the constraint that $p_i^{(j)}(\hat{x}_{i-1}) = p_{i-1}^{(j)}(\hat{x}_{i-1})$, $j = 0, 1, \ldots,$ SMTH, then $\|f - p_i\|_{\hat{S}} \leq$ TOL.

Again, if we can find such an $\hat{x}_{i-1}$ then the algorithm is successfully terminated. If not, the algorithm is terminated and an appropriate error message is generated.

<u>Remarks</u>. Since any $p_i(x)$ can be written $p_i(x) = \sum_{j=0}^{N-1} a_j(x - x_{i-1})^j$, it is a simple matter to meet the smoothness constraints. In fact, it is easy to generalize this scheme in order to force p_i to satisfy Hermite interpolatory constraints at several points. Hence, it would be easy to modify the above algorithm so that we could interpolate the function being approximated and its derivatives at the knots if desired.

In our implementation of this algorithm we have modularized the approximation routine so that we may use a Remes-type algorithm for computing uniform approximation subject to interpolating constraints [1], and we use Householder transforms to compute the discrete L^2 approximations.

Note that when using uniform approximations computed by the Remes algorithm it is in general not necessary to compute the best approximation on a particular subinterval in order to conclude that the subinterval is too long. Indeed, at each iteration of the Remes algorithm we obtain a lower bound for the error of the best approximation on this particular subinterval. Consequently, if during some iteration of the Remes algorithm this lower bound is greater than TOL, we may conclude that the present subinterval is too large and terminate the Remes algorithm.

In our implementation of this algorithm, the $\tilde{x}_i$ are chosen as follows. At each step of this iterative procedure we will let $\tilde{a}$ be the current largest point in X such that requirements (1) and (2) of the appropriate algorithm are satisfied on $[x_{i-1}, \tilde{a}] \cap X$, and we will let $\tilde{b}$ be the current smallest point in X such that $\tilde{b} > \tilde{a}$ and requirement (2) of the appropriate algorithm fails to be satisfied. We initialize this process by computing (or attempting to compute) the appropriate best approximation on $[x_{i-1}, b] \cap X$. If this approximation satisfies requirement (2), then we set $\tilde{x}_i = b$ and we are done. If the approximation fails to satisfy (2), we set $\tilde{b} = b$. Next, let $t = \min\{x \in X\colon\ [x_{i-1}, x] \cap X$ contains at least max(2, N-SMTH) points$\}$. If the approximation on $[x_{i-1}, t] \cap X$ fails to satisfy (2) then the algorithm cannot meet the desired accuracy and fails. Otherwise, we set $\tilde{a} = t$.

In general, we proceed as follows. We let $t = \inf\{x \in X\colon\ (\tilde{b} - \tilde{a})/2 \leq x < \tilde{b}\}$. If this set is empty, we set $t = \sup\{x \in X\colon\ \tilde{a} \leq x \leq (\tilde{b} - \tilde{a})/2\}$. If $t = \tilde{a}$ then this procedure has converged and we set $\tilde{x}_i = t = \tilde{a}$. Otherwise, we compute (or attempt to compute) the appropriate approximation on $[x_{i-1}, t] \cap X$. If this

approximation satisfies (2) then we set $\tilde{a} = t$. If this approximation fails to satisfy (2) then we set $\tilde{b} = t$. We continue this process until $\tilde{b} - \tilde{a}$ is less than some user definable prescribed tolerance, at which point we accept $\tilde{a}$ as a good approximation to $\tilde{x}_i$ and terminate this procedure. We compute the $\hat{x}_i$ in a manner analogous to the above.

In an attempt to accelerate the convergence of the above scheme when using the best uniform approximation operator, we have tried to take advantage of the fact that corresponding to each $\tilde{a}$ we know the error of approximation on $[x_{i-1}, \tilde{a}] \cap X$, call it SMLERR, and corresponding to each $\tilde{b}$ we know a lower bound for the error of approximation on $[x_{i-1}, \tilde{b}] \cap X$, call it BIGERR. We change the above scheme by first setting $\alpha = (\text{BIGERR} - \text{TOL})/(\text{BIGERR} - \text{SMLERR})$ and then setting $t = \inf\{x \in X: \alpha\tilde{a} + (1 - \alpha)\tilde{b} \leq x < \tilde{b}\}$ or, if this set is empty, we set $t = \sup\{x \in X: \tilde{a} \leq x \leq \alpha\tilde{a} + (1 - \tilde{\alpha})\tilde{b}\}$ in the general iteration described above. Hence, if SMLERR is very close to TOL, t will be chosen close to $\tilde{a}$. Our numerical experience has shown that this procedure only works well when approximating uniformly smooth functions and that this algorithm cannot be significantly improved by allowing the Remes algorithm to run to completion in order to obtain the true error of approximation for BIGERR.

III. Numerical Results

This algorithm has been implemented as FORTRAN programs and has been tested on Colorado State University's CDC CYBER 172. In the appendix we give a listing of the algorithm using the best uniform approximation operator. By using a least squares routine in place of the REMES subroutine (and those subroutines called only by REMES: SOLVE, DIVIDF, ZEROFD, BPOLY, TRANS) the least squares version of this algorithm is readily obtained. Indeed, we essentially replaced these six subroutines by four subroutines: ASET, HOUSE, TOLCHK and FIX. ASET sets up the overdetermined system to be solved in the L^2 sense. Here a certain

amount of care needs to be taken to insure that the approximation on each succeeding interval has the desired smoothness at the common endpoints. Thus, if we are currently considering the interval $[x_{i-1}, \tilde{x}_i]$ and p_{i-1} is the approximation that was accepted for $[x_{i-2}, x_{i-1}]$ then we form the following system:

$$\sum_{\nu=k+1}^{N-1} c_{\nu+1}(x-x_{i-1})^{\nu} = f(x) - \sum_{\nu=0}^{k} \frac{p_{i-1}^{(\nu)}(x_{i-1})}{\nu!}(x-x_{i-1})^{\nu} \text{ for } \begin{cases} x \in [x_{i-1}, \tilde{x}_i] \cap X \text{ if } k = -1 \\ x \in (x_{i-1}, \tilde{x}_i] \cap X \text{ if } k > -1 \end{cases}$$

where k = SMTH = smoothness desired at x_{i-1} and $\sum\limits_{\nu=0}^{-1}$ is to be replaced by 0 whenever it occurs. HOUSE is any good overdetermined least squares solver. We use one based on Householder transforms (a simple version since our system will always be nondegenerate). TOLCHK compares the maximum absolute value of f minus the approximation returned by HOUSE with TOL and also finds the set of $N - k - 1$ last "extreme" values in $[x_{i-1}, \tilde{x}_i]$. That is, points at which $|f(x) - p(x)|$ is a local maximum relative to points on each side of x. This is for the backing off from $\tilde{x}_i$ procedure to increase stability. FIX simply converts the coefficients $\{c_\nu\}$ of a polynomial of the form $\sum\limits_{\nu=1}^{N} c_\nu (x-x_{i-1})^{\nu-1}$ into coefficients $\{b_\nu\}$ for the same polynomial written in the form $\sum\limits_{\nu=1}^{N} b_\nu x^{\nu-1}$.

As examples, using the best uniform operator the functions $e^{|x|}$ on $[-1,1]$, $|\sin(x)|$ on $[-\pi,\pi]$ and $\sqrt{x}$ on $[0,2]$ were approximated on 200 equally spaced points with TOL = .01, SMTH = 2, and N = 6. Each of these examples is difficult to approximate by polynomials near x = 0. Consequently, the algorithm's ability to automatically decrease the length of the subintervals near x = 0 and then recover by lengthening them for x > 0 is tested. Below is a table listing knot locations (subinterval endpoints) and the CPU time in seconds used to compute the piecewise polynomial approximation.

	$e^{\|x\|}$	$\|\sin(x)\|$	$\sqrt{x}$
CPU TIME	.979	1.095	.876
Knot Locations	-1.0	-3.14	0.0
	- .347	- .268	.0804
	- .0653	.0474	.231
	.0452	.205	.352
	.196	.458	.814
	.437	.868	2.0
	1.0	1.53	-----
	-----	3.14	-----

Because uniform approximations weight each data point equally, they are particularly suited for approximating precise mathematical functions or for approximating data with noise levels that are very small relative to the desired accuracy. However, when these algorithms are used to approximate functions containing considerable levels of noise, in general the approximations tend to follow the noise patterns more than is desirable so that near abrupt changes in the data, the approximations tend to begin to oscillate and generally it requires several subintervals to dampen these oscillations. (Some counterexamples from engineering applications have been found which show that this is not always the case.)

For example, we were given some experimental data involving the release of bitumen from oil shale heated to a constant temperature as a function of time. Because relatively few data points were available, we filled in the gaps between the data points by linear interpolation so that we approximated on 200 equally spaced points. Figure 1 is a plot of the data being approximated and the approximation generated using uniform approximation with N = 6, SMTH = 2, and TOL = 2.5.

Notice that as oscillations appear in the data between times of 22 and 34 minutes, greatly exaggerated oscillations appear in the approximation.

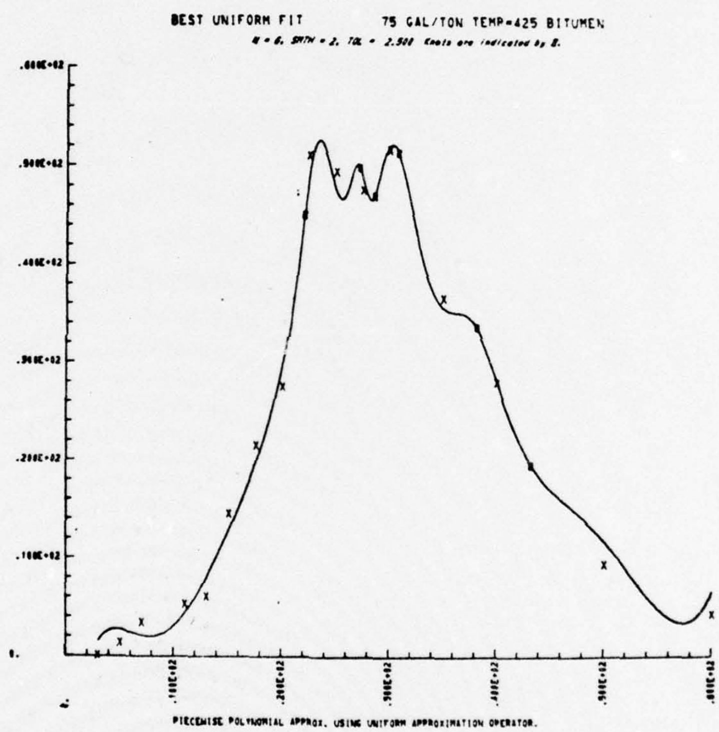


Figure 1

Because L^2 approximations minimize the effects of normally distributed random noise, we expect the L^2 local approximation operator to be prefereable for this type of example. Indeed, in Figure 2 we present a plot of the same data using the algorithm with the L^2 operator.

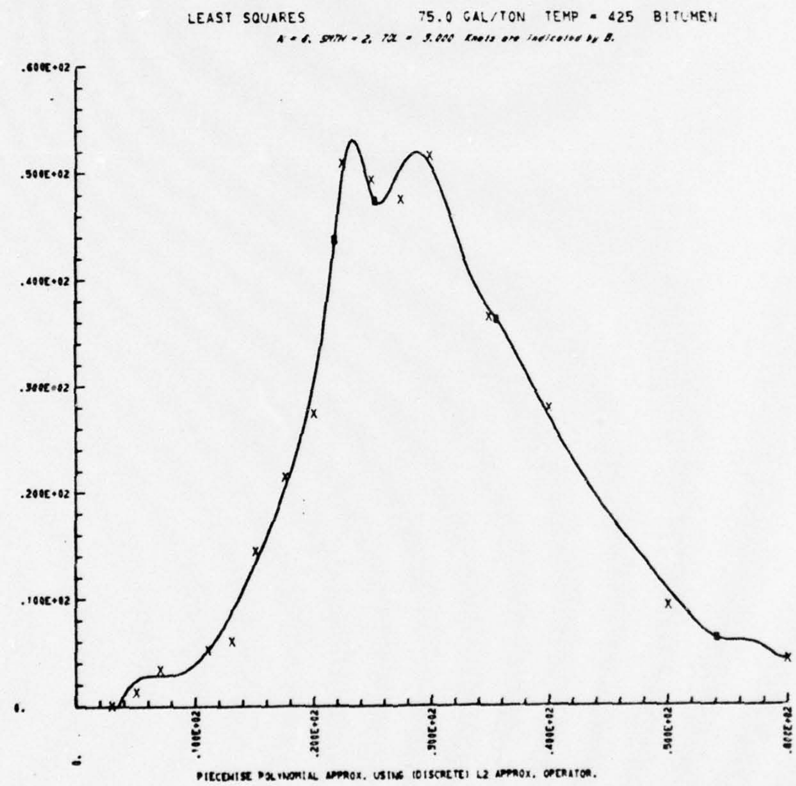


Figure 2

If oscillations do tend to occur in the approximation, often one can improve the approximation by approximating on more (densely packed) points. In particular, by adding extra points in regions where the function being approximated is unstable, and removing them in regions where the function is "nice", for the reason outlined above, the approximations tend to improve, with only moderately increased computational costs.

For some applications it might be preferable to be able to choose ahead of time a particular knot location and to specify interpolatory constraints at these knots. For example, when approximating $|\sin(x)|$ it might be advantageous to force a knot to be located at $x = 0$ and to require $p(0) = 0$, $p'(0) = -1$ as x approaches 0 from the left, and $p'(0) = +1$ as x approaches 0 from the right. The modifications to the algorithm as given necessary to accomplish this would not be extensive or difficult.

It should also be remarked that this algorithm can be used with linear approximating families other than polynomials. For example, any generalized Haar subspace of high enough order (to allow Hermite interpolating constraints) can be used with the best uniform approximation operator. In addition, one could force not only knots to occur at various prescribed points but could also vary the smoothness at these knots as well as change the form of the approximating families at these knots.

Recently, John Rice has described an adaptive piecewise polynomial algorithm which uses local Hermite interpolation instead of uniform approximation on each subinterval to obtain each polynomial piece [2]. His algorithm requires that the first SMTH derivatives or approximate derivatives of f be available in order to compute approximations which have SMTH continuous derivatives. Rice's adaptive strategy also differs from ours; he uses a bisection strategy which can be described as follows. First, compute the Hermite interpolating polynomial to f on [a, b]. That is, compute the polynomial which interpolates f and its first SMTH derivatives at both endpoints, as well as interpolating f at k evenly distributed points in (a, b), where k = DEGREE - 2·SMTH - 1, and DEGREE is the degree of the approximating polynomial. Next, measure the error of approximation using any preselected L^p norm ($p \geq 1$). If the error is less that TOL, the algorithm terminates, otherwise bisect [a, b] into two subintervals and check to see if the Hermite interpolating polynomial on [a, (a + b)/2] differs from f in norm by

no more than TOL. Meanwhile, place the subinterval [(a + b)/2, b] in a "stack" to be processed later. Continue bisecting subintervals, placing the right half of the interval on the top of the stack to be processed later and check the left half to see if the error of approximation by the Hermite interpolating polynomial is less than TOL. When a subinterval and its associated approximations are found which meet the desired tolerance, we accept this approximation as a "piece" of the piecewise polynomial approximation and continue the algorithm by removing the subinterval which is currently at the top of the stack and repeating the above on it unless the stack is empty, at which point terminate the algorithm.

Rice's routine requires the value of the function and its derivatives at very many points in [a, b], even though it may not actually access these values. His algorithm is particularly suited, then, for approximating precise mathematical functions for which this data is readily available. Our algorithm seems more suitable for obtaining approximations of functions given in tabular form (as in many engineering applications) since our algorithm does not require any information about derivatives, and no special configuation of the data is needed for our adaptive strategy to work. See the forthcoming paper by Andrews, Hull and Taylor for further comments on suitability of this algorithm for this type of application.

By using interpolatory constraints on both endpoints of each subinterval, and interpolating the values of the derivative of f at these points, we could modify our algorithm to use a bisection adaptive strategy. However, in order to take greatest advantage of our use of the best approximation operators as opposed to the Hermite interpolation operator, we felt that it was preferable to use the majority of our degrees of freedom inside subintervals instead of using them up on interpolation requirements at the endpoints. Essentially, in our algorithm we traded a somewhat faster run time (since computing best approximations is very slow relative to computing interpolatory polynomials) for generally fewer knots and the freedom from needing to know derivative information.

REFERENCES

[1] H.L. Loeb, D.G. Moursund, L.L. Schumaker, G.D.Taylor, Uniform Generalized Weight Function Polynomial Approximation with Interpolation, SIAM J. Numer. Anal., Vol. 6, No. 2, June, 1969.

[2] J.R. Rice, An Algorithm for Adaptive Piecewise Polynomial Approximation, to appear.

APPENDIX

Here we give listings for the two algorithms described in this paper with the uniform approximation listing first and the least squares approximation listing second. In both cases we include a driver and sample output where the output corresponds to figure 1 and figure 2, respectively. Also, we list the input of these two runs between the listing of the code and the output.

```
      PROGRAM DRIVER(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
      LOGICAL EPROR
      COMMON XTABLE(500),FTABLE(500),DUMMY(1382)
      INTEGER SMTH
      READ (5,100) N,SMTH,TOL
      WRITE (6,200) N,SMTH,TOL
      MAXNUM=0
   10 MAXNUM=MAXNUM+1
      READ (5,150) XTABLE(MAXNUM),FTABLE(MAXNUM)
      IF (EOF(5).EQ.0.0) GO TO 10
      MAXNUM=MAXNUM-1
      CALL LINEAR(MAXNUM,200)
      MAXNUM=200
      CALL UNIACF(TOL,N,SMTH,MAXNUM,ERROR)
  100 FORMAT (2I5,F10.5)
  150 FORMAT (2F10.5)
  200 FORMAT (1H1,47HALGORITHM USING UNIFORM APPROXIMATION OPERATOR.,/,
     $4HON =,I3,7H SMTH =,I3,6H TOL =,F10.5)
      END


      SUBROUTINE LINEAR(OLDMAX,NEWMAX)
C
C  WE USE THIS SUBROUTINE TO FILL IN BETWEEN THE ORIGINAL DATA POINTS
C  BY LINEAR INTERPOLATION WHEN THERE ARE TOO FEW POINTS FOR THE EF-
C  FECTIVE USE OF THE ALGORITHM.  OLDMAX IS THE NUMBER OF ORIGINAL
C  DATA POINTS, NEWMAX IS THE TOTAL NUMBER OF INTERPOLATED DATA POINTS
C  DESIRED.  THE NEW NEWMAX POINTS ARE EQUALLY SPACED BETWEEN
C  XTABLE(1) AND XTABLE(OLDMAX).
C
      INTEGER OLDMAX
      COMMON XTABLE(500),FTABLE(1582),X(150),Y(150)
      DO 10 I=1,OLDMAX
         X(I)=XTABLE(I)
         Y(I)=FTABLE(I)
   10 CONTINUE
      DELTA=(X(OLDMAX)-X(1))/FLOAT(NEWMAX-1)
      K=1
      DO 40 I=2,NEWMAX
         XX=XTABLE(1)+FLOAT(I-1)*DELTA
         XTABLE(I)=XX
   20    IF (XX.LE.X(K+1)) GO TO 30
         K=K+1
         GO TO 20
   30    FTABLE(I)=Y(K)+(XX-X(K))*(Y(K+1)-Y(K))/(X(K+1)-X(K))
   40 CONTINUE
      RETURN
C
      END


      SUBROUTINE UNIACF(TOL,N,SMTH,MAXNUM,ERROR)
C
C  THIS SUBROUTINE ADAPTIVELY COMPUTES A PIECEWISE POLYNOMIAL APPROX-
C  IMATION OF DEGREE N-1 TO THE FUNCTION STORED IN THE ARRAYS XTABLE
C  AND FTABLE WITH SMTH CONTINUOUS DERIVATIVES AND WHICH DEVIATES FROM
C  THIS FUNCTION BY NO MORE THAN TOL AT ANY POINT IN XTABLE.
C
C  THE PARAMETERS ARE AS FOLLOWS--
C
C     N - THE NUMBER OF COEFFICIENTS OF EACH POLYNOMIAL PIECE--I.E.,
C         ONE MORE THAN THE DEGREE OF THE PIECEWISE POLYNOMIAL APPROXI-
```

```
C        MATION.  N MUST BE GREATER THAN 1. AND AS THE ARRAYS ARE CUR-          UNIAC120
C        RENTLY DIMENSIONED, IT IS ASSUMED THAT N IS NO BIGGER THAN 17.         UNIAC130
C                                                                               UNIAC140
C     SMTH - THE NUMBER OF CONTINUOUS DERIVATIVES DESIRED OF THE                UNIAC150
C        APPROXIMATION.  SMTH MUST NOT BE GREATER THAN N-2.  IF ONLY            UNIAC160
C        CONTINUITY OF THE APPROXIMATION IS REQUIRED, SET SMTH = 0.             UNIAC170
C        (IN FACT AN APPROXIMATION WHICH IS DISCONTINUOUS AT THE KNOTS          UNIAC180
C        MAY BE OBTAINED BY SETTING SMTH = -1.)                                 UNIAC190
C                                                                               UNIAC200
C     MAXNUM - THE NUMBER OF POINTS ACTUALLY STORED IN THE ARRAYS               UNIAC210
C        XTABLE AND FTABLE.  (AS THESE ARRAYS ARE CURRENTLY DIMENSIONED         UNIAC220
C        MAXNUM MUST BE LESS THAN 500.  IF ONE WISHES TO COMPUTE AP-            UNIAC230
C        PROXIMATIONS TO FUNCTIONS WITH MORE THAN 500 POINTS, ONE CAN           UNIAC240
C        EASILY MODIFY THIS PROGRAM TO CONTINUOUSLY READ IN MORE POINTS         UNIAC250
C        AFTER ROOM IS MADE IN THE TWO STORAGE ARRAYS, XTABLE AND               UNIAC260
C        FTABLE BY HAVING COMPLETED SEVERAL SUBINTERVALS.)                       UNIAC270
C                                                                               UNIAC280
C     TOL - THE TOLERANCE THAT THE APPROXIMATION MUST SATISFY.                  UNIAC290
C                                                                               UNIAC300
C     ERROR - A LOGICAL VALUE SET TO .TRUE. IF AN ERROR OCCURS IN THE           UNIAC310
C        PROGRAM (AN APPROPRIATE ERROR MESSAGE WILL ALSO BE PRINTED)            UNIAC320
C        AND SET TO .FALSE. OTHERWISE.                                          UNIAC330
C                                                                               UNIAC340
C     BLANK COMMON PROVIDES THE REMAINDER OF THE INPUT.  IT IS ASSUMED          UNIAC350
C     THAT THE FIRST 500 WORDS OF BLANK COMMON CONTAIN THE TABLE OF X           UNIAC360
C     VALUES (XTABLE), AND THAT THE NEXT 500 WORDS CONTAIN THE TABLE OF         UNIAC370
C     FUNCTION VALUES (FTABLE).  THE NEXT TWO WORDS IN BLANK COMMON ARE         UNIAC380
C     USED INTERNALLY THROUGHOUT THE PROGRAM.  THE NEXT 1080 WORDS OF           UNIAC390
C     BLANK COMMON STORAGE CONTAIN THE COMPUTED COEFFICIENTS AND THE            UNIAC400
C     KNOTS--I.E., CSTORE(I,INT) IS THE COEFFICIENT OF THE I-1ST DEGREE         UNIAC410
C     TERM IN SUBINTERVAL NUMBER INT.  CSTORE(N+1,INT) IS THE LEFT END          UNIAC420
C     POINT OF SUBINTERVAL INT.  THE LAST ARRAY OF 300 WORDS IS USED            UNIAC430
C     INTERNALLY TO STORE THE ERRORS OF APPROXIMATION OCCURING AT EACH          UNIAC440
C     DATA POINT DURING THE REMES ALGORITHM, AND IT IS ALSO USED AS A           UNIAC450
C     SCRATCH STORAGE ARRAY THROUGHOUT THE PROGRAM.                             UNIAC460
C                                                                               UNIAC470
C  THE COEFFICIENTS AND SUBINTERVAL ENDPOINTS ARE PRINTED OUT AS THEY           UNIAC480
C  ARE COMPUTED.  ALSO THE FUNCTION EVAL IS AVAILABLE TO THE USER TO            UNIAC490
C  EVALUATE THE APPROXIMATION AT ANY POINT WITHIN THE ENTIRE INTERVAL.          UNIAC500
C                                                                               UNIAC510
      LOGICAL LAST,ERROR,DONE,ABORT                                             UNIAC520
      INTEGER SMTH                                                              UNIAC530
      DIMENSION C(18)                                                           UNIAC540
      COMMON XTABLE(1000),LCTNLE,LCTNRE,DUM1(1380)                              UNIAC550
      COMMON /SCALAR/ NPLUS0,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT           UNIAC560
      DATA MAXINT/301/,NRSMTH/-1/                                               UNIAC570
C                                                                               UNIAC580
      ERROR=.FALSE.                                                             UNIAC590
      DONE=.FALSE.                                                              UNIAC600
      ABORT=.FALSE.                                                             UNIAC610
      NPLUS0=N                                                                  UNIAC620
      NPLUS1=N+1                                                                UNIAC630
C                                                                               UNIAC640
C  IN THE FIRST SUBINTERVAL THERE ARE NO INTERPOLATORY CONSTRAINTS--            UNIAC650
C  CONSEQUENTLY, WE SET NLSMTH=-1.                                              UNIAC660
C                                                                               UNIAC670
      NLSMTH=-1                                                                 UNIAC680
      NX=NPLUS1-2-NLSMTH-NRSMTH                                                 UNIAC690
      NXM1=NX-1                                                                 UNIAC700
      LNGTH=NX+1                                                                UNIAC710
C                                                                               UNIAC720
C  WE INITIALLY TRY AS MUCH OF THE CURRENT REMAINING PORTION OF THE             UNIAC730
```

```
C   WHOLE INTERVAL AS POSSIBLE AS AN INITIAL GUESS FOR EACH SUCCESSIVE
C   SUBINTERVAL.  LCTNLE IS THE LOCATION (IN THE ARRAY XTABLE) OF THE
C   LEFT END POINT OF THE CURRENT SUBINTERVAL, LCTNRE IS THE LOCATION OF
C   THE RIGHT END POINT.
C
      LCTNLE=1
      LCTNRE=MIN0(MAXNUM,MAXINT)
      DO 20 INTNUM=1,60
         NINT=INTNUM
C
C   SUBROUTINE COMPUT FINDS THE LARGEST SUBINTERVAL OF (XTABLE(LCTNLE),
C   XTABLE(LCTNRE)) WITH LEFT END POINT XTABLE(LCTNLE) SUCH THAT THE BEST
C   APPROXIMATION ON THIS SUBINTERVAL SATISFIES ALL THE CONSTRAINTS.  THE
C   RIGHT END POINT IS #BACKED OFF# TO THE LAST INTERIOR EXTREME POINT
C   OF F-P TO ADD TO THE STABILITY OF THE ALGORITHM.  THE LOCATION OF
C   THIS RIGHT END POINT IS STORED IN LCTNRE.  IF LCTNRE=MAXNUM (I.E.
C   WE ARE DONE), CONTROL IS PASSED TO LINE 40.  IF NO SUCH SUBINTERVAL
C   CAN BE FOUND, CONTROL IS PASSED TO LINE 50.  IF THERE ARE FEWER THAN
C   LNGTH POINTS FROM LCTNRE TO MAXNUM, LAST IS SET TO .TRUE. AND THE
C   SPECIAL CASE SUBROUTINE LSTINT IS CALLED.
C
         CALL COMPUT (C,TOL,LNGTH,MAXNUM,LAST,DONE,ABORT)
         IF (ABORT) GO TO 50
         IF (DONE) GO TO 40
         IF (INTNUM.GT.1) GO TO 10
         NLSMTH=SMTH
         NX=NPLUS1-2-NLSMTH-NRSMTH
         NXM1=NX-1
         LNGTH=NX+1
   10    IF (LAST) GO TO 30
C
C   SUBROUTINE STORE STORES THE COEFFICIENTS FOR THIS SUBINTERVAL IN THE
C   ARRAY CSTORE.  IT ALSO PRINTS OUT THE COEFFICIENTS AND THE ERROR OF
C   APPROXIMATION ON THIS SUBINTERVAL.  SUBROUTINE SETP(C,X,K) STORES THE
C   VALUE OF THE POLYNOMIAL DETERMINED BY THE COEFFICIENTS IN THE ARRAY
C   C AND ITS FIRST K DERIVATIVES AT THE POINT X IN THE ARRAY PPRIME.
C
         CALL STORE (C,LCTNLE,LCTNRE)
         CALL SETP (C,XTABLE(LCTNRE),NLSMTH)
         LCTNLE=LCTNRE
         LCTNRE=MIN0(MAXNUM,MAXINT+LCTNLE-1)
   20 CONTINUE
      WRITE (6,60) NINT
      ERROR=.TRUE.
      RETURN
   30 CALL LSTINT (C,TOL,LNGTH,MAXNUM,ABORT)
      IF (ABORT) GO TO 50
   40 CALL STORE (C,LCTNLE,LCTNRE)
      RETURN
   50 WRITE (6,70)
      ERROR=.TRUE.
      RETURN
C
   60 FORMAT (1H0,39(2H* ),1H*,/,2H0*,12X, 37HTHIS APPROXIMATION REQUIRE
     1S MORE THAN,I3, 13H SUBINTERVALS,12X,1H*,/,2H0*,28X, 20H--PROGRAM
     2ABORTING--,29X,1H*,/,1H0,39(2H* ),1H*)
   70 FORMAT (1H0,39(2H* ),1H*,/,2H0*,15X, 46HTHE ALGORITHM CANNOT MEET
     1THE DESIRED ACCURACY,16X,1H*,/,2H0*,28X, 20H--PROGRAM ABORTING--,2
     29X,1H*,/,1H0,39(2H* ),1H*)
C
      END
```

```
      SUBROUTINE COMPUT(C,TOL,LNGTH,MAXNUM,LAST,DONE,ABORT)
C
C  THIS SUBROUTINE FINDS THE LARGEST SUBINTERVAL AND THE BEST APPROX-
C  IMATION TO F ON THIS SUBINTERVAL SUCH THAT THE APPROXIMATION MEETS
C  THE DESIRED ERROR TOLERANCE ON THE SUBINTERVAL.
C
      INTEGER A,B
      LOGICAL LAST,OK,DONE,ABORT,TOOBIG
      REAL C(18)
      COMMON XTABLE(1000),LCTNLE,LCTNRE,CSTORE(18,60),CDERIV(300)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      COMMON /COMP/ LCTNX(18)
C
C  WE ASSUME THAT WE ARE CLOSE ENOUGH TO THE TRUE LARGEST SUBINTERVAL
C  RIGHT END POINT WHEN WE KNOW THAT OUR APPROXIMATION TO THE TRUE RIGHT
C  END POINT IS WITHIN ETA OF THE TRUE END POINT.
C
      DATA ETA/.08/
C
      LITTLE=LCTNLE+LNGTH-1
      A=0
      LAST=.FALSE.
   10 NUMPTS=LCTNRE-LCTNLE+1
C
C  IF THE ACCURACY OF THE BEST APPROXIMATION ON THE CURRENT SUBINTERVAL
C  EXCEEDS TOL, CONTROL IS PASSED TO LINE 30.
C
      CALL REMES (C,LCTNX,TOL,TOOBIG,ABORT)
      IF (ABORT) RETURN
      IF (TOOBIG) GO TO 30
      IF (LCTNRE.LT.MAXNUM) GO TO 20
      DONE=.TRUE.
      RETURN
C
C  A IS THE CURRENT LARGEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT
C  THE BEST APPROXIMATION ON THIS SUBINTERVAL SATIFIES ALL CONSTRAINTS.
C
   20 A=LCTNRE
      IF ((XTABLE(B)-XTABLE(A).GT.ETA).AND.(B-A.GT.1)) GO TO 40
      GO TO 50
C
C  B IS THE CURRENT SMALLEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT
C  THE BEST APPROXIMATION ON THIS SUBINTERVAL FAILS TO SATISFY THE CON-
C  STRAINTS.
C
   30 B=LCTNRE
   40 NEWTRY=(A+B)/2+1
      IF (NEWTRY.EQ.B) NEWTRY=NEWTRY-1
      IF (NEWTRY.LT.LITTLE) NEWTRY=LITTLE
      IF (NEWTRY.EQ.LCTNRE) GO TO 50
      LCTNRE=NEWTRY
      GO TO 10
C
C  IF A IS STILL 0, THEN NO SUBINTERVAL WITH AT LEAST LNGTH POINTS
C  WILL WORK, SO THE ALGORITHM IS TERMINATED.
C
   50 IF (A.NE.0) GO TO 60
      ABORT=.TRUE.
      RETURN
C
C  SINCE NEWTRY IS ALWAYS STRICTLY LESS THAN THE CURRENT B, IF NEWTRY=
C  LCTNRE, AND A IS NOT STILL 0, NEWTRY=A, WHICH IS A POINT WHICH SAT-
```

```
C  ISFIES ALL REQUIREMENTS.  WE NOW BACK THE RIGHT ENDPOINT OFF TO THE
C  BEST INTERIOR EXTREME POINT OF F-P TO ADD TO THE STABILITY OF THE AL-
C  GORITHM.
C
   60 DO 70 I=1,N
         CDERIV(I)=C(I)
         CSTORE(I,NINT)=C(I)
   70 CONTINUE
      NDUMMY=N
      CALL DERIV (CDERIV,NDUMMY)
      I=NX
      LCTNRE=LCTNX(I)
      NEWTRY=LCTNRE
      SMLL=CMPR(CDERIV,NDUMMY,NEWTRY+LCTNLE-1,OK)
      IF (OK) GO TO 100
   80 I=I-1
      IF (I.EQ.0) GO TO 100
      NEWTRY=LCTNX(I)
      IF (NEWTRY.LT.LNGTH) GO TO 100
      TEMP=CMPR(CDERIV,NDUMMY,NEWTRY+LCTNLE-1,OK)
      IF (.NOT.OK) GO TO 90
      LCTNRE=NEWTRY
      GO TO 100
   90 IF (TEMP.GE.SMLL) GO TO 80
      SMLL=TEMP
      LCTNRE=NEWTRY
      GO TO 80
  100 LCTNRE=LCTNLE+LCTNRE-1
      IF (MAXNUM-LCTNRE+1.LT.LNGTH) LAST=.TRUE.
      RETURN
C
      END


      SUBROUTINE LSTINT(C,TOL,LNGTH,MAXNUM,ABORT)
C
C  THIS SUBROUTINE HANDLES THE SPECIAL CASE OF FINDING A SUBINTERVAL
C  AND A BEST APPROXIMATION ON THAT SUBINTERVAL WHEN THERE ARE TOO
C  FEW REMAINING POINTS FOR COMPUT TO WORK.
C
      INTEGER OLDLE,OLDRE
      LOGICAL TOOBIG,ABORT
      REAL C(18)
      COMMON X(1000),LCTNLE,LCTNRE,CSTORE(18,60)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      COMMON /COMP/ LCTNX(18)
C
      DO 10 OLDLE=1,NPLUS1
   10 CSTORE(OLDLE,NINT)=C(OLDLE)
      OLDLE=LCTNLE
      OLDRE=LCTNRE
      LCTNRE=MAXNUM
      LCTNLE=MIN0(MAXNUM-LNGTH+1,(MAXNUM-OLDLE+1)/2)-1
   20 LCTNLE=LCTNLE+1
      IF (MAXNUM-LCTNLE+1.LT.LNGTH) GO TO 40
      CALL SETP (CSTORE(1,NINT),X(LCTNLE),NLSMTH)
      NUMPTS=LCTNRE-LCTNLE+1
      CALL REMES (C,LCTNX,TOL,TOOBIG,ABORT)
      IF (ABORT) RETURN
      IF (TOOBIG) GO TO 20
      CALL STORE (CSTORE(1,NINT),OLDLE,LCTNLE)
      NINT=NINT+1
```

```
      DO 30 OLDLE=1,NPLUS1
   30 CSTORE(OLDLE,NINT)=C(OLDLE)
      RETURN
   40 ABORT=.TRUE.
      RETURN
C
      END


      SUBROUTINE STORE(C,LCTNLE,LCTNRE)
C
C  THIS SUBROUTINE OUTPUTS THE COEFFICIENTS AND ENDPOINTS OF THE
C  CURRENT APPROXIMATION AND SUBINTERVAL.  APPROPRIATE INFORMATION
C  IS STORED IN THE ARRAY CSTORE TO ALLOW THE ENTIRE PIECEWISE POLY-
C  NOMIAL APPROXIMATION TO BE EASILY EVALUATED AT ANY POINT BY THE
C  FUNCTION EVAL.
C
      DIMENSION C(18)
      COMMON XTABLE(500),FTABLE(502),CSTORE(18,60),DUM1(300)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
C
      NUMPTS=LCTNRE-LCTNLE+1
      WRITE (6,30) NINT,XTABLE(LCTNLE),XTABLE(LCTNRE),NUMPTS
      WRITE (6,40) (I,C(I),I=1,N)
      ERR=0.0
      DO 10 I=LCTNLE,LCTNRE
         TEMP=ABS(FTABLE(I)-HORNER(C,XTABLE(I),N))
         IF (TEMP.GT.ERR) ERR=TEMP
   10 CONTINUE
      WRITE (6,50) ERR
      DO 20 I=1,N
   20 CSTORE(I,NINT)=C(I)
      CSTORE(NPLUS1,NINT)=XTABLE(LCTNLE)
      RETURN
C
   30 FORMAT (//,5X, 15HINTERVAL NUMBER,I4, 16H WHICH BEGINS AT,E23.16,/
     1, 12H AND ENDS AT,E23.16,2X,  8HCONTAINS,I4,  8H POINTS.,/, 60H TH
     2E COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE,/)
   40 FORMAT ((10X,  2HC(,I2,  3H) =,E24.16))
   50 FORMAT (/, 47H THE ERROR OF APPROXIMATION IN THIS INTERVAL IS,E24.
     116,  1H.)
C
      END


      SUBROUTINE DERIV(C,N)
C
C  THIS SUBROUTINE REPLACES THE COEFFICIENTS OF A POLYNOMIAL IN STAND-
C  ARD FORM WITH THE COEFFICIENTS OF THIS POLYNOMIAL-S DERIVATIVE.
C  THE NUMBER OF COEFFICIENTS, N, IS DECREMENTED.
C
      DIMENSION C(N)
C
      N=N-1
      DO 10 I=1,N
   10 C(I)=FLOAT(I)*C(I+1)
      RETURN
C
      END


      SUBROUTINE SETP(C,X,SMTH)
```

```
C
C  THIS SUBROUTINE APPROPRIATELY STORES IN THE ARRAY PPRIME THE VAL-
C  UES WHICH MUST BE INTERPOLATED TO GIVE THE PIECEWISE POLYNOMIAL THE
C  DESIRED SMOOTHNESS.
C
      DIMENSION C(18)
      COMMON /COMP/ CSTORE(18)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      COMMON /DDIF/ PPRIME(5),DUM1(18)
      INTEGER SMTH
C
      DO 10 I=1,N
   10 CSTORE(I)=C(I)
      NDUMMY=N
      I=0
   20 IF (I.GT.SMTH) RETURN
      IF (I.EQ.0) GO TO 30
      CALL DERIV (CSTORE,NDUMMY)
   30 PPRIME(I+1)=HORNER(CSTORE,X,NDUMMY)
      I=I+1
      GO TO 20
C
      END


      FUNCTION CMPR(C,N,NEWTRY,OK)
C
C  THIS SUBROUTINE COMPARES THE FIRST DERIVATIVE OF THE CURRENT PIECE OF
C  THE PIECEWISE POLYNOMIAL APPROXIMATION EVALUATED AT XTABLE(NEWTRY)
C  WITH THE FIRST DERIVATIVE OF THE QUADRATIC INTERPOLATION OF F, CEN-
C  TERED AROUND XTABLE(NEWTRY), EVALUATED AT XTABLE(NEWTRY).  IF THESE
C  TWO DIFFER IN ABSOLUTE VALUE BY LESS THAN TOLER (EITHER ABSOLUTELY
C  OR RELATIVELY), WE SET OK TO .TRUE. AND WE ACCEPT XTABLE(NEWTRY) AS A
C  REASONABLE SUBINTERVAL RIGHT ENDPOINT.  NOTE THAT THE USER MAY EASILY
C  CHANGE TOLER BY MEANS OF THE FOLLOWING DATA STATEMENT.
C
      LOGICAL OK
      COMMON X(500),F(500),DUM1(1382)
      DIMENSION C(18)
      DATA TOLER/.05/
C
      OK=.FALSE.
      A=(F(NEWTRY)-F(NEWTRY-1))/(X(NEWTRY)-X(NEWTRY-1))
      B=(F(NEWTRY+1)-F(NEWTRY))/(X(NEWTRY+1)-X(NEWTRY))
      D=(B-A)/(X(NEWTRY+1)-X(NEWTRY-1))
      CMPR=A+D*(X(NEWTRY)-X(NEWTRY-1))
      A=TOLER*CMPR
      CMPR=ABS(CMPR-HORNER(C,X(NEWTRY),N))
      IF (CMPR.LE.A) OK=.TRUE.
      RETURN
C
      END


      SUBROUTINE REMES(C,LCTNX,TOL,TOOBIG,ABORT)
C
C  THIS IS THE DRIVING PROGRAM FOR THE COMPUTATION OF THE BEST
C  UNIFORM POLYNOMIAL APPROXIMATION TO F(X) (VALUES ARE STORED
C  IN XTABLE AND FTABLE) OF DEGREE LESS THAN OR EQUAL TO N-1 ON
C  THE SUBINTERVAL [XTABLE(LCTNLE),XTABLE(LCTNRE)].  SEE INTRODUCTION TO
C  APPROXIMATION THEORY BY E. W. CHENEY FOR A COMPLETE DISCUSSION OF
C  THIS ALGORITHM.
```

```
C
      DIMENSION C(18),SGNRXI(18),LCTNX(18),LCTNZ(18)
      COMMON XTABLE(500),FTABLE(500),LCTNLE,DUMMY(1081),ERROR(300)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMGRD,NINT
      LOGICAL STOP,TOOBIG,ABORT
      INTEGER FRSTM1,START
C
C  EPS IS A MACHINE CONSTANT--SET EPS TO (VERY ROUGHLY) THE SMALLEST
C  VALUE SUCH THAT EPS + 1.0 IS GREATER THAN 1.0.
C
      DATA ITERMX,EPS/9,1.0E-10/
C
C  FIRST WE INITIALIZE VARIOUS ARRAYS AND VARIABLES.
C
      TOOBIG=.FALSE.
      FRSTM1=LCTNLE-1
      SGNRXI(1)=1.0
      DO 10 I=2,NX
   10 SGNRXI(I)=-SGNRXI(I-1)
      NFPTS=NUMGRD-2
      START=2
      ERROR(1)=0.0
      IF (NLSMTH.GE.0) GO TO 20
      NFPTS=NFPTS+1
      START=1
   20 IF (NRSMTH.LT.0) NFPTS=NFPTS+1
      DELTA=FLOAT(NFPTS-1)/FLOAT(NX-1)
      DO 30 I=1,NX
   30 LCTNX(I)=START+IFIX(FLOAT(I-1)*DELTA+.5)
C
C  NOW WE BEGIN ITERATING.  CONVERGENCE OCCURS WHEN TWO CONSECUTIVE
C  REFERENCE SETS (DETERMINED BY SOLVE) ARE THE SAME.
C
      DO 80 ITER=1,ITERMX
         CALL DIVDIF (C,LCTNX,SGNRXI,ABORT)
         IF (ABS(C(NPLUS1)).GT.TOL) TOOBIG=.TRUE.
         IF (ABORT.OR.TOOBIG) RETURN
         DO 40 I=START,NUMGRD
   40    ERROR(I)=FTABLE(I+FRSTM1)-BPOLY(XTABLE(I+FRSTM1),C,N)
         IF (ABS(C(NPLUS1)).LE.EPS) GO TO 60
         SGNRXI(1)=SIGN(1.0,ERROR(LCTNX(1)))
         DO 50 I=2,NX
            SGNRXI(I)=SIGN(1.0,ERROR(LCTNX(I)))
            IF (SGNRXI(I)*SGNRXI(I-1).GE.0.0) GO TO 90
   50    CONTINUE
   60    CALL ZEROFD (LCTNX,LCTNZ,SGNRXI,ERROR)
         CALL SOLVE (LCTNX,LCTNZ,SGNRXI,TOL,C(NPLUS1),STOP,TOOBIG)
         IF (C(NPLUS1).GT.TOL) STOP=TOOBIG=.TRUE.
         IF (.NOT.STOP) GO TO 80
         IF (.NOT.TOOBIG) CALL TRANS (C,N)
         RETURN
   80 CONTINUE
C
C  THE REMAINING STATEMENTS ARE FOR DEBUGGING.
C
      WRITE (6,110) ITERMX
      GO TO 100
   90 WRITE (6,120) ITER
  100 ABORT=.TRUE.
      RETURN
C
  110 FORMAT (1H0,39(2H* ),1H*,/,2H0*,11X, 40HTHE REMES ALGORITHM HAS NO
```

```
      1T CONVERGED IN,I3, 12H ITERATIONS.,11X,1H*,/,2H0*,11X, 36HPROGRAM   REMES710
     2ABORTED IN SUBROUTINE REMES.,30X,1H*,/,1H0,39(2H* ),1H*)           REMES720
  120 FORMAT (1H0,39(2H* ),1H*,/,2H0*,8X, 12HIN ITERATION,I3, 47H OF REM REMES730
     1ES, NO ALTERNATION OF SIGN OCCURS AT THE,7X,1H*,/,2H0*,8X, 57HEXTR REMES740
     2EME POINTS.  THE PROGRAM ABORTED IN SUBROUTINE REMES.,12X,1H*,/,1H REMES750
     30,39(2H* ),1H*)                                                     REMES760
C                                                                         REMES770
      END                                                                 REMES780


      SUBROUTINE DIVDIF(C,LCTNX,SGNRXI,ABORT)                             DIVDIF10
C                                                                         DIVDIF20
C  THIS SUBROUTINE MAKES USE OF A DIVIDED DIFFERENCE SCHEME FOR           DIVDIF30
C  SOLVING THE VANDERMONDE-LIKE LINEAR SYSTEM INHERENT IN THE REMES       DIVDIF40
C  ALGORITHM.  THE ADVANTAGES OF USING THIS SPECIAL PURPOSE LINEAR        DIVDIF50
C  SYSTEM SOLVER ARE--                                                    DIVDIF60
C                                                                         DIVDIF70
C     THIS ROUTINE REQUIRES ON THE ORDER OF N**2 OPERATION AS COM-        DIVDIF80
C     PARED TO GAUSSIAN ELIMINATION WHICH REQUIRES ON THE ORDER           DIVDIF90
C     OF N**3 OPERATIONS.                                                 DIVDI100
C                                                                         DIVDI110
C     THIS ROUTINE REQUIRES ON THE ORDER OF N STORAGE LOCATIONS           DIVDI120
C     AS COMPARED TO GAUSSIAN ELIMINATION WHICH REQUIRES ON THE           DIVDI130
C     ORDER OF N**2.                                                      DIVDI140
C                                                                         DIVDI150
C  SEE THE FORTHCOMING PAPER BY J.A. HULL, S.F. MCCORMICK, AND G.D.       DIVDI160
C  TAYLOR FOR A COMPLETE DESCRIPTION OF THIS ALGORITHM.                   DIVDI170
C                                                                         DIVDI180
      COMMON XTABLE(500),FTABLE(500),LCTNLE,LCTNRE,CSTORE(18,60),         DIVDI190
     1ERROR(300)                                                          DIVDI200
      COMMON /DDIF/ PPRIME(5),X(18)                                       DIVDI210
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT          DIVDI220
      DIMENSION LCTNX(18),C(18),SGNRXI(18),D(18)                          DIVDI230
      INTEGER FRSTM1                                                      DIVDI240
      LOGICAL ABORT                                                       DIVDI250
C                                                                         DIVDI260
C  FIRST WE INITIALIZE SEVERAL VARIABLES.                                 DIVDI270
C                                                                         DIVDI280
      FRSTM1=LCTNLE-1                                                     DIVDI290
      IF (NRSMTH.GE.0) SGNRXI(NPLUS1)=0.0                                 DIVDI300
      IF (NRSMTH.LT.0) SGNRXI(NPLUS1)=SGNRXI(NX)                          DIVDI310
C                                                                         DIVDI320
C  SET UP THE VECTOR X AND THE FIRST ROW OF THE DIVIDED DIFFERENCE TAB-   DIVDI330
C  LE, USING THE COEFFICIENT VECTOR C FOR TEMPORARY STORAGE.              DIVDI340
C                                                                         DIVDI350
      I=0                                                                 DIVDI360
   10 IF (I.GT.NLSMTH) GO TO 20                                           DIVDI370
      I=I+1                                                               DIVDI380
      X(I)=XTABLE(LCTNLE)                                                 DIVDI390
      C(I)=PPRIME(1)                                                      DIVDI400
      GO TO 10                                                            DIVDI410
   20 J=0                                                                 DIVDI420
   30 IF (J.GE.NX) GO TO 40                                               DIVDI430
      I=I+1                                                               DIVDI440
      J=J+1                                                               DIVDI450
      X(I)=XTABLE(FRSTM1+LCTNX(J))                                        DIVDI460
      C(I)=FTABLE(FRSTM1+LCTNX(J))                                        DIVDI470
      GO TO 30                                                            DIVDI480
   40 IF (I.GE.NPLUS1) GO TO 50                                           DIVDI490
      I=I+1                                                               DIVDI500
      X(I)=XTABLE(LCTNRE)                                                 DIVDI510
      C(I)=PPRIME(NLSMTH+2)                                               DIVDI520
```

```
      GO TO 40                                                          DIVDI530
   50 CONTINUE                                                          DIVDI540
C                                                                       DIVDI550
C  WE NOW COMPUTE THE NEEDED DIVIDED DIFFERENCES.                       DIVDI560
C                                                                       DIVDI570
      FAC=1.0                                                           DIVDI580
      DO 100 J=2,N                                                      DIVDI590
         JP1=J+1                                                        DIVDI600
         JM1=J-1                                                        DIVDI610
         FAC=FAC*FLOAT(JM1)                                             DIVDI620
         TEMP2=C(JM1)                                                   DIVDI630
         DO 90 I=J,N                                                    DIVDI640
            IF (X(I).NE.X(I-JM1)) GO TO 70                              DIVDI650
            IF (I.GT.NLSMTH+1) GO TO 60                                 DIVDI660
            IF (J.GT.NPLUS1-NX) GO TO 220                               DIVDI670
            TEMP1=PPRIME(J)/FAC                                         DIVDI680
            GO TO 80                                                    DIVDI690
   60       IF (NLSMTH+JP1.GT.NPLUS1-NX) GO TO 220                      DIVDI700
            TEMP1=PPRIME(NLSMTH+JP1)/FAC                                DIVDI710
   70       TEMP1=(C(I)-C(I-1))/(X(I)-X(I-JM1))                         DIVDI720
   80       C(I-1)=TEMP2                                                DIVDI730
            TEMP2=TEMP1                                                 DIVDI740
   90    CONTINUE                                                       DIVDI750
         C(N)=TEMP2                                                     DIVDI760
  100 CONTINUE                                                          DIVDI770
C                                                                       DIVDI780
C  L**(-1)F HAS NOW BEEN TEMPORARILY STORED IN THE COEFFICIENT ARRAY C. DIVDI790
C  WE NOW COMPUTE L**(-1)C.  WE WILL COMPUTE THE DIVIDED DIFFERENCES    DIVDI800
C  IN THE TEMPORARY STORAGE ARRAY D.  FIRST WE SET UP THE FIRST COLUMN  DIVDI810
C  OF THE DIVIDED DIFFERENCE TABLE.                                     DIVDI820
C                                                                       DIVDI830
      I=0                                                               DIVDI840
  110 IF (I.GT.NLSMTH) GO TO 120                                        DIVDI850
      I=I+1                                                             DIVDI860
      D(I)=0.0                                                          DIVDI870
      GO TO 110                                                         DIVDI880
  120 J=0                                                               DIVDI890
  130 IF (J.GE.NX) GO TO 140                                            DIVDI900
      I=I+1                                                             DIVDI910
      J=J+1                                                             DIVDI920
      D(I)=SGNRXI(J)                                                    DIVDI930
      GO TO 130                                                         DIVDI940
  140 IF (I.GE.N) GO TO 150                                             DIVDI950
      I=I+1                                                             DIVDI960
      D(I)=0.0                                                          DIVDI970
      GO TO 140                                                         DIVDI980
  150 CONTINUE                                                          DIVDI990
C                                                                       DIVD1000
C  WE NOW COMPUTE THE NEEDED DIVIDED DIFFERENCES.                       DIVD1010
C                                                                       DIVD1020
      DO 190 J=2,N                                                      DIVD1030
         TEMP2=D(J-1)                                                   DIVD1040
         DO 180 I=J,N                                                   DIVD1050
            IF (X(I).NE.X(I-J+1)) GO TO 160                             DIVD1060
            TEMP1=0.0                                                   DIVD1070
            GO TO 170                                                   DIVD1080
  160       TEMP1=(D(I)-D(I-1))/(X(I)-X(I-J+1))                         DIVD1090
  170       D(I-1)=TEMP2                                                DIVD1100
            TEMP2=TEMP1                                                 DIVD1110
  180    CONTINUE                                                       DIVD1120
         D(N)=TEMP2                                                     DIVD1130
  190 CONTINUE                                                          DIVD1140
```

```
C
C     WE NOW COMPUTE M=(F(N+1)-W[TRANSPOSE]*B1))/(0.0-W[TRANSPOSE]*B2)
C     =C(N+1)=(UNIFORM ERROR)
C     FIRST WE COMPUTE THE TWO DOT PRODUCTS SIMULTANEOUSLY.
C
      W=1.0
      B2=0.0
      B1=B2
      DO 200 I=1,N
         B1=B1+C(I)*W
         B2=B2+D(I)*W
         W=W*(X(NPLUS1)-X(I))
  200 CONTINUE
C
C     NOW WE COMPUTE M
C
      C(NPLUS1)=(C(NPLUS1)-B1)/(SGNRXI(NPLUS1)-B2)
C
C     FINALLY, WE COMPUTE THE COEFFICIENTS C(I).
C
      DO 210 I=1,N
         C(I)=C(I)-C(NPLUS1)*D(I)
  210 CONTINUE
      RETURN
  220 ABORT=.TRUE.
      WRITE (6,230)
      RETURN
C
  230 FORMAT (1H0,39(2H* ),1H*,/,2H0*,11X, 54HSUBROUTINE DIVDIF HAS FAIL
     1ED--PROBABLY DUE TO AN INPUT,12X,1H*,/,2H0*,11X, 58HERROR (TO DIVD
     2IF)--NOT ENOUGH ELEMENTS IN THE ARRAY PPRIME,8X,1H*,/,2H0*,11X, 56
     3HOR TWO IDENTICAL POINTS IN THE ARRAY X.  PROGRAM ABORTED,10X,1H*,
     4/,2H0*,11X, 21HIN SUBROUTINE DIVDIF.,45X,1H*,/,1H0,39(2H* ),1H*)
C
      END


      SUBROUTINE ZEROFD(LCTNX,LCTNZ,SGNRXI,ERROR)
C
C   THIS SUBROUTINE LOCATES THE (APPROXIMATE) ZEROS BETWEEN CONSECUTIVE
C   POINTS OF THE CURRENT REFERENCE SET.  THE LOCATIONS OF THESE POINTS
C   IN THE ARRAY XTABLE RELATIVE TO THE BEGINNING OF THE CURRENT SUB-
C   INTERVAL ARE STORED IN LCTNZ.
C
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMGRD,NINT
      DIMENSION LCTNX(18),LCTNZ(18),SGNRXI(18),ERROR(300)
C
      LCTNZ(1)=MIN0(2,2+NLSMTH)
      LCTNZ(NX+1)=MAX0(NUMGRD-1,NUMGRD-1-NRSMTH)
      DO 30 I=2,NX
         LTNIM1=LCTNX(I-1)
         NUMPTS=LCTNX(I)-LTNIM1
         DO 10 J=1,NUMPTS
            NEWTRY=LTNIM1+J
            IF (ERROR(LTNIM1)*ERROR(NEWTRY).LE.0.0) GO TO 20
   10    CONTINUE
   20    LCTNZ(I)=NEWTRY
   30 CONTINUE
      RETURN
C
      END
```

```
      SUBROUTINE SOLVE(LCTNX,LCTNZ,SGNRXI,TOL,SMALL,STOP,TOOBIG)
C
C  THIS SUBROUTINE PERFORMS THE MULTIPLE EXCHANGE OF THE REFERENCE
C  SET REQUIRED AT EACH ITERATION OF THE REMES ALGORITHM.  SEE
C  INTRODUCTION TO APPROXIMATION THEORY BY E. W. CHENEY FOR A COM-
C  PLETE DISCUSSION OF THIS ALGORITHM.
C
      COMMON XTABLE(1000),LCTNLE,DUMMY(1081),ERROR(300)
      COMMON /SCALAR/ N,NPLUS1,NX,NXM1,DUM(4)
      DIMENSION LCTNX(18),LCTNZ(18),SGNRXI(18)
      LOGICAL STOP,TOOBIG
      INTEGER FRSTM1,RTEND
C
C  EPS AND TWOEPS ARE MACHINE CONSTANTS--SET EPS (VERY ROUGHLY) TO THE
C  SMALLEST NUMBER SUCH THAT 1.0 + EPS IS GREATER THAN 1.0, AND TWOEPS
C  TO 2*EPS.  IT IS NOT CRITICAL THAT THESE VALUES BE PRECISE.
C
      DATA EPS,TWOEPS/5.0E-11,1.0E-10/
C
      STOP=.TRUE.
      FRSTM1=LCTNLE-1
C
C  WE FIRST COMPUTE THE LOCATIONS OF THE NEW SET OF EXTREME POINTS,
C  STORING THEM IN THE VECTOR LCTNX.  WE BEGIN BY CHOOSING AS THE ITH
C  ELEMENT OF LCTNX THE LOCATION OF THE GRIDPOINT IN THE SUBINTERVAL
C  BETWEEN THE ITH AND (I+1)ST ZERO WHICH RESULTS IN THE LARGEST ERROR
C  OF THE SAME SIGN AS THE PREVIOUS ITH EXTREME POINT (THEREBY GUARANT-
C  EEING ALTERNATION).  AT THE SAME TIME WE SEARCH FOR THE GRIDPOINT
C  WHICH RESULTS IN THE LARGEST (ABSOLUTE) ERROR, STORING ITS LOCATION
C  (IN LNBGST) AND THE NUMBER OF THE SUBINTERVAL IN WHICH IT OCCURS (IN
C  INBGST).
C
      BIGER=EPS
      BIGEST=EPS
      DO 30 INTNUM=1,NX
         BIG=EPS
         LFTEND=LCTNZ(INTNUM)
         RTEND=LCTNZ(INTNUM+1)
         SGN=SGNRXI(INTNUM)
         DO 20 NEWLOC=LFTEND,RTEND
            TEMP=SGN*ERROR(NEWLOC)
            IF (TEMP.LE.BIG) GO TO 10
            LNBG=NEWLOC
            BIG=TEMP
   10       TEMP=ABS(TEMP)
            IF (TEMP.LE.BIGEST) GO TO 20
            BIGEST=TEMP
            LNBGST=NEWLOC
            INBGST=INTNUM
   20    CONTINUE
         IF(INTNUM.EQ.1)SMALL=BIG
         IF (BIG.LT.TWOEPS) GO TO 30
         IF (BIG.GT.BIGER) BIGER=BIG
         IF (BIG.LT.SMALL) SMALL=BIG
         IF (LCTNX(INTNUM).EQ.LNBG) GO TO 30
         LCTNX(INTNUM)=LNBG
         STOP=.FALSE.
   30 CONTINUE
      IF (BIGEST.LT.TWOEPS) RETURN
      IF (SMALL.LE.TOL) GO TO 40
      STOP=.TRUE.
      TOOBIG=.TRUE.
```

```
      RETURN
   40 IF (ABS(BIGER-BIGEST).LT.EPS) RETURN
C
C  AT THIS POINT IT IS STILL NECESSARY TO INSERT THE LOCATION OF THE
C  GRIDPOINT RESULTING IN THE LARGEST ERROR INTO LCTNX.  THERE ARE THREE
C  CASES. EACH OF WHICH IS HANDLED SEPARATELY--ALTERNATION IS PRESERVED
C  AT THE GRIDPOINTS.
C
      STOP=.FALSE.
      YBIGST=XTABLE(FRSTM1+LNBGST)
      YIBIG=XTABLE(LCTNX(INBGST)+FRSTM1)
      IF ((INBGST.EQ.1).AND.(YBIGST.LT.YIBIG)) GO TO 50
      IF ((INBGST.EQ.NX).AND.(YBIGST.GT.YIBIG)) GO TO 70
      IF (YBIGST.LT.YIBIG) NEWINT=INBGST-1
      IF (YBIGST.GT.YIBIG) NEWINT=INBGST+1
      LCTNX(NEWINT)=LNBGST
      RETURN
   50 DO 60 I=2,NX
         J=NX-I+2
         LCTNX(J)=LCTNX(J-1)
   60 CONTINUE
      LCTNX(1)=LNBGST
      RETURN
   70 DO 80 J=1,NXM1
         LCTNX(J)=LCTNX(J+1)
   80 CONTINUE
      LCTNX(NX)=LNBGST
      RETURN
C
      END


      FUNCTION BPOLY(XX,C,N)
C
C  THIS FUNCTION IS USED TO EVALUATE (BY THE APPROPRIATE ADAPTATION
C  OF HORNERS METHOD) THE POLYNOMIAL
C
C     C(1) + C(2)*(XX-X(1)) + C(3)*(XX-X(1))*(XX-X(2)) + . . .
C       + C(N)*(XX-X(1))*(XX-X(2))* . . . *(XX-X(N-1))
C
      DIMENSION C(18)
      COMMON /DDIF/ DUMMY(5),X(18)
C
      NM1=N-1
      BPOLY=C(N)
      DO 10 I=1,NM1
         J=N-I
         BPOLY=C(J)+(XX-X(J))*BPOLY
   10 CONTINUE
      RETURN
C
      END


      SUBROUTINE TRANS(C,N)
C
C  THIS SUBROUTINE TRANSFORMS A POLYNOMIAL WRITTEN IN THE FORM
C
C     C(1) + C(2)*(X-X(1)) + C(3)*(X-X(1))*(X-X(2)) + . . .
C       + C(N)*(X-X(1))*(X-X(2))* . . . *(X-X(N-1))
C
C  TO A POLYNOMIAL WRITTEN IN TERMS OF POWERS OF X.  THE X(I)-S
```

```
C  ARE SUPPLIED BY SUBROUTINE DIVDIF.
C
      DIMENSION C(18)
      COMMON /DDIF/ DUMMY(5),X(18)
C
      NM1=N-1
      DO 20 J=1,NM1
         K=N-J
         DO 10 I=K,NM1
            C(I)=C(I)-X(K)*C(I+1)
   10    CONTINUE
   20 CONTINUE
      RETURN
C
      END


      FUNCTION EVAL(X)
C
C  THIS FUNCTION EVALUATES THE PIECEWISE POLYNOMIAL APPROXIMATION
C  AT ANY POINT IN THE ENTIRE INTERVAL.
C
      COMMON DUMMY(1002),CSTORE(18,60),DUM(300)
      COMMON /SCALAR/ N,NPLUS1,DUM2(5),NINT
C
      IF (NINT.LT.2) GO TO 20
      DO 10 I=2,NINT
         ISTORE=I-1
         IF (X.LE.CSTORE(NPLUS1,I)) GO TO 30
   10 CONTINUE
      ISTORE=NINT
      GO TO 30
   20 ISTORE=1
   30 EVAL=HORNER(CSTORE(1,ISTORE),X,N)
      RETURN
C
      END


      FUNCTION HORNER(C,X,N)
C
C  THIS FUNCTION EVALUATES A POLYNOMIAL IN STANDARD FORM BY HORNERS
C  METHOD.
C
      DIMENSION C(N)
C
      HORNER=C(N)
      I=N
   10 IF (I.LT.2) RETURN
      HORNER=HORNER*X+C(I-1)
      I=I-1
      GO TO 10
C
      END
```

UNIFORM ADAPTIVE CURVE FITTING PROGRAM : SAMPLE RUN

INPUT :

```
        6     2     2.50        ( N, SMTH, TOL )

        3.0         0.0         (XTABLE, FTABLE)
        5.0         1.3
        7.0         3.4
       11.0         5.2
       13.0         6.0
       15.0        14.4
       17.5        21.4
       20.0        27.4
       22.5        50.9
       25.0        49.3
       27.5        47.5
       30.0        51.5
       35.0        36.5
       40.0        27.9
       50.0         9.4
       60.0         4.2
```

OUTPUT :

```
     INTERVAL NUMBER  1 WHICH BEGINS AT   .3000000000000000E+01
AND ENDS AT   .2190452261306518E+02  CONTAINS  67 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

          C( 1) = -.1634025000676115E+02
          C( 2) =  .1181130531149097E+02
          C( 3) = -.2637146272918883E+01
          C( 4) =  .2593743838385461E+00
          C( 5) = -.1129567576998175E-01
          C( 6) =  .1865450590541398E-03

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS   .1969452589748130E+01.


     INTERVAL NUMBER  2 WHICH BEGINS AT   .2190452261306518E+02
AND ENDS AT   .2706030150753747E+02  CONTAINS  19 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

          C( 1) =  .7960326211065352E+06
          C( 2) = -.1656777306495747E+06
          C( 3) =  .1375385536550829E+05
          C( 4) = -.5692778474236584E+03
          C( 5) =  .1174880789994893E+02
          C( 6) = -.9672838852406418E-01

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS   .2294874459374114E+01.


     INTERVAL NUMBER  3 WHICH BEGINS AT   .2706030150753747E+02
AND ENDS AT   .2849246231155757E+02  CONTAINS   6 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE
```

```
C( 1) = -.4555522308418542E+07
C( 2) =  .7815330984931737E+06
C( 3) = -.5356190391737269E+05
C( 4) =  .1833051913127594E+04
C( 5) = -.3132567622722183E+02
C( 6) =  .2138556370651710E+00
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .2350077767167932E+01.

INTERVAL NUMBER 4 WHICH BEGINS AT .2849246231155757E+02
AND ENDS AT .3078391959798978E+02 CONTAINS 9 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) =  .1040230085838158E+07
C( 2) = -.1649042600538041E+06
C( 3) =  .1043144021371787E+05
C( 4) = -.3291270419910106E+03
C( 5) =  .5179689635444362E+01
C( 6) = -.3252942438648798E-01
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .2350077744816190E+01.

INTERVAL NUMBER 5 WHICH BEGINS AT .3078391959798978E+02
AND ENDS AT .3794472361809017E+02 CONTAINS 26 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) = -.2329820690203374E+06
C( 2) =  .3289571488642064E+05
C( 3) = -.1849503807458532E+04
C( 4) =  .5177682249693407E+02
C( 5) = -.7218440333123937E+00
C( 6) =  .4009674821594894E-02
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .2347864915846003E+01.

INTERVAL NUMBER 6 WHICH BEGINS AT .3794472361809017E+02
AND ENDS AT .4310050251256257E+02 CONTAINS 19 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) = -.6827434677450825E+05
C( 2) =  .7613513411310967E+04
C( 3) = -.3373191291063522E+03
C( 4) =  .7428809770578283E+01
C( 5) = -.8135813032576111E-01
C( 6) =  .3545453066058774E-03
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .2416020625049441E+01.

INTERVAL NUMBER 7 WHICH BEGINS AT .4310050251256257E+02
AND ENDS AT .5999999999999955E+02 CONTAINS 60 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) =  .1333616589900339E+05
C( 2) = -.1192965606757301E+04
C( 3) =  .4193870667632291E+02
C( 4) = -.7183879075542343E+00
C( 5) =  .5934796153467731E-02
C( 6) = -.1861388516033458E-04
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .2416020624874818E+01.

```
      PROGRAM DRIVER(INPUT,TAPE5=INPUT,OUTPUT,TAPE6=OUTPUT)
      COMMON XTABLE(500),FTABLE(500),DUMMY(1382)
      INTEGER SMTH
      READ(5,20)N,SMTH,TOL
      WRITE(6,30)
      WRITE(6,40)N,SMTH,TOL
      MAXNUM=0
   10 MAXNUM=MAXNUM+1
      READ(5,50)XTABLE(MAXNUM),FTABLE(MAXNUM)
      IF(EOF(5).EQ.0.0)GO TO 10
      MAXNUM=MAXNUM-1
      CALL LINEAR(MAXNUM,400)
      MAXNUM=400
      CALL L2ACF(TOL,N,SMTH,MAXNUM,ERROR)
C
   20 FORMAT(2I5,F10.5)
   30 FORMAT(1H1,5X,48HPIECEWISE POLYNOMIAL APPROXIMATION--L2 OPERATOR.)
   40 FORMAT(1H0,5X,17HNUMBER OF COEFFS=,I2,1H,,I2,25H CONTINUOUS DERIVS
     1, TOL= ,F7.5)
   50 FORMAT(2F10.5)
C
      END


      SUBROUTINE LINEAR(OLDMAX,NEWMAX)
C
C  THIS SUBROUTINE FILLS IN BETWEEN THE ORIGINAL DATA POINTS BY LINEAR
C  INTERPOLATION WHEN THERE ARE TOO FEW POINTS FOR THE EFFECTIVE USE
C  OF THE ALGORITHM.  OLDMAX IS THE NUMBER OF ORIGINAL DATA POINTS.
C  NEWMAX IS THE TOTAL NUMBER OF INTERPOLATED DATA POINTS DESIRED.
C  THE NEWMAX POINTS ARE EQUALLY SPACED BETWEEN XTABLE(1) AND
C  XTABLE(OLDMAX).  DUE TO THIS FACT, THE ORIGINAL DATA POINTS MAY NOT
C  BE IN XTABLE AND FTABLE UPON COMPLETION OF THIS ROUTINE.  THUS,
C  NEWMAX MUST BE CHOSEN SUFFICIENTLY LARGE (LESS THAN OR EQUAL TO 500)
C  IN ORDER TO RETAIN THE PROXIMITIES OF THE ORIGINAL DATA POINTS.
C
      INTEGER OLDMAX
      COMMON XTABLE(500),FTABLE(1582),X(150),Y(150)
C
      DO 10 I=1,OLDMAX
        X(I)=XTABLE(I)
        Y(I)=FTABLE(I)
   10 CONTINUE
      DELTA=(X(OLDMAX)-X(1))/FLOAT(NEWMAX-1)
      K=1
      DO 40 I=2,NEWMAX
        XX=XTABLE(1)+FLOAT(I-1)*DELTA
        XTABLE(I)=XX
   20   IF (XX.LE.X(K+1)) GO TO 30
        K=K+1
        GO TO 20
   30   FTABLE(I)=Y(K)+(XX-X(K))*(Y(K+1)-Y(K))/(X(K+1)-X(K))
   40 CONTINUE
      RETURN
C
      END


      SUBROUTINE L2ACF(TOL,N,SMTH,MAXNUM,ERROR)
C
C  THIS SUBROUTINE ADAPTIVELY COMPUTES A PIECEWISE POLYNOMIAL APPROX-
C  IMATION OF DEGREE N-1 TO THE FUNCTION STORED IN THE ARRAYS XTABLE
```

```
C  AND FTABLE WITH SMTH CONTINUOUS DERIVATIVES.  THE APPROXIMATION DEV-
C  IATES FROM THIS FUNCTION BY NO MORE THAN TOL AT ANY POINT IN XTABLE.
C
C  THE PARAMETERS ARE AS FOLLOWS--
C
C     N - THE NUMBER OF COEFFICIENTS OF EACH POLYNOMIAL PIECE--IE. ONE
C        MORE THAN THE DEGREE OF THE PIECEWISE POLYNOMIAL APPROXIMATION.
C        N MUST BE GREATER THAN 1, AND AS THE ARRAYS ARE CURRENTLY
C        DIMENSIONED, IT IS ASSUMED THAT N IS NO BIGGER THAN 17.
C
C     SMTH - THE NUMBER OF CONTINUOUS DERIVATIVES DESIRED OF THE APPROX-
C        IMATION.  SMTH MUST NOT BE GREATER THAN N-2.  IF ONLY CONTIN-
C        UITY OF THE APPROXIMATION IS DESIRED, SET SMTH = 0.  AN APPROX-
C        IMATION WHICH IS DISCONTINUOUS AT THE KNOTS MAY BE OBTAINED BY
C        SETTING SMTH = -1.
C
C     MAXNUM - THE NUMBER OF POINTS ACTUALLY STORED IN THE ARRAYS XTABLE
C        AND FTABLE.  AS THESE ARRAYS ARE CURRENTLY DIMENSIONED, MAXNUM
C        MUST BE LESS THAN OR EQUAL TO 500.
C
C     TOL - THE TOLERANCE THAT THE APPROXIMATION MUST SATISFY.
C
C     ERROR - A LOGICAL VALUE SET TO .TRUE. IF AN ERROR OCCURS IN THE
C        PROGRAM (AN APPROPRIATE ERROR MESSAGE WILL ALSO BE PRINTED)
C        AND SET TO .FALSE. OTHERWISE.
C
C  THE COEFFICIENTS AND SUBINTERVAL ENDPOINTS ARE PRINTED OUT AS THEY
C  ARE COMPUTED.  ALSO THE FUNCTION EVAL IS AVAILABLE TO THE USER TO
C  EVALUATE THE APPROXIMATION AT ANY POINT WITHIN THE ENTIRE INTERVAL.
C
      LOGICAL LAST,ERROR,DONE,ABORT
      INTEGER SMTH
      DIMENSION C(18)
      COMMON XTABLE(1000),LCTNLE,LCTNRE,DUM1(1380)
      COMMON/SCALAR/NPLUS0,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      DATA MAXINT/150/,NRSMTH/-1/
C
      ERROR=.FALSE.
      DONE=.FALSE.
      ABORT=.FALSE.
      NPLUS0=N
      NPLUS1=N+1
C
C  IN THE FIRST SUBINTERVAL THERE ARE NO INTERPOLATORY CONSTRAINTS--
C  CONSEQUENTLY, WE SET NLSMTH=-1.
C
      NLSMTH=-1
      NX=N-2-NLSMTH-NRSMTH
      NXM1=NX-1
      LNGTH=NX+1
C
C  WE INITIALLY TRY AS MUCH OF THE CURRENT REMAINING PORTION OF THE
C  WHOLE INTERVAL AS POSSIBLE AS AN INITIAL GUESS FOR EACH SUCCESSIVE
C  SUBINTERVAL.  LCTNLE IS THE LOCATION (IN THE ARRAY XTABLE) OF THE
C  LEFT ENDPOINT OF THE CURRENT SUBINTERVAL, LCTNRE IS THE LOCATION
C  OF THE RIGHT ENDPOINT.
C
      LCTNLE=1
      LCTNRE=MINO(MAXNUM,MAXINT)
      DO 20 INTNUM=1,60
         NINT=INTNUM
C
```

```
C  SUBROUTINE COMPUT FINDS THE LARGEST SUBINTERVAL OF (XTABLE(LCTNLE),
C  XTABLE(LCTNRE)) WITH LEFT ENDPOINT XTABLE(LCTNLE) SUCH THAT THE BEST
C  APPROXIMATION ON THIS SUBINTERVAL SATISFIES ALL THE CONSTRAINTS.
C  THE RIGHT ENDPOINT IS -BACKED OFF- TO THE LAST INTERIOR EXTREME
C  POINT OF F-P TO ADD TO THE STABILITY OF THE ALGORITHM.  THE LOCATION
C  OF THIS RIGHT ENDPOINT IS STORED IN LCTNRE.  IF LCTNRE=MAXNUM (I.E.,
C  WE ARE DONE), CONTROL IS PASSED TO LINE 40.  IF NO SUCH SUBINTERVAL
C  CAN BE FOUND, CONTROL IS PASSED TO LINE 50.  IF THERE ARE FEWER THAN
C  LNGTH POINTS FROM LCTNRE TO MAXNUM, LAST IS SET TO .TRUE. AND THE
C  SPECIAL CASE SUBROUTINE LSTINT IS CALLED.
C
         CALL COMPUT(C,TOL,LNGTH,MAXNUM,LAST,DONE,ABORT)
         IF (DONE) GO TO 40
         IF (ABORT) GO TO 50
         IF (INTNUM.GT.1) GO TO 10
         NLSMTH=SMTH
         NX=N-2-NLSMTH-NRSMTH
         NXM1=NX-1
         LNGTH=NX+1
   10    IF (LAST) GO TO 30
C
C  SUBROUTINE STORE STORES THE COEFFICIENTS FOR THIS SUBINTERVAL IN THE
C  ARRAY CSTORE.  IT ALSO PRINTS OUT THE COEFFICIENTS AND THE ERROR OF
C  APPROXIMATION ON THIS SUBINTERVAL.  SUBROUTINE SETP(C,X,K) STORES THE
C  VALUE OF THE POLYNOMIAL DETERMINED BY THE COEFFICIENTS IN THE ARRAY
C  C AND ITS FIRST K DERIVATIVES AT THE POINT X IN THE ARRAY PPRIME.
C
         CALL STORE(C,LCTNLE,LCTNRE)
         CALL SETP(C,XTABLE(LCTNRE),NLSMTH)
         LCTNLE=LCTNRE
         LCTNRE=MIN0(MAXNUM,MAXINT+LCTNLE-1)
   20 CONTINUE
      WRITE (6,60) NINT
      ERROR=.TRUE.
      RETURN
   30 CALL LSTINT (C,TOL,LNGTH,MAXNUM,ABORT)
      IF (ABORT) GO TO 50
   40 CALL STORE (C,LCTNLE,LCTNRE)
      RETURN
   50 WRITE (6,70)
      ERROR=.TRUE.
      RETURN
C
   60 FORMAT (1H0,39(2H* ),1H*,/,2H0*,12X, 37HTHIS APPROXIMATION REQUIRE
     1S MORE THAN,I3, 13H SUBINTERVALS,12X,1H*,/,2H0*,28X, 20H--PROGRAM
     2ABORTING--,29X,1H*,/,1H0,39(2H* ),1H*)
   70 FORMAT (1H0,39(2H* ),1H*,/,2H0*,15X, 46HTHE ALGORITHM CANNOT MEET
     1THE DESIRED ACCURACY,16X,1H*,/,2H0*,28X, 20H--PROGRAM ABORTING--,2
     29X,1H*,/,1H0,39(2H* ),1H*)
C
      END


      SUBROUTINE STORE(C,LCTNLE,LCTNRE)
C
C  THIS SUBROUTINE OUTPUTS THE COEFFICIENTS AND ENDPOINTS OF THE CUR-
C  RENT APPROXIMATION AND SUBINTERVAL.  APPROPRIATE INFORMATION IS
C  STORED IN THE ARRAY CSTORE TO ALLOW THE ENTIRE PIECEWISE POLYNOMIAL
C  APPROXIMATION TO BE EASILY EVALUATED AT ANY POINT BY THE FUNCTION
C  EVAL.
C
      DIMENSION C(18)
```

```
      COMMON XTABLE(500),FTABLE(502),CSTORE(16,60),DUM1(300)
      COMMON/SCALAR/N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
C
      NUMPTS=LCTNRE-LCTNLE+1
      WRITE (6,30) NINT,XTABLE(LCTNLE),XTABLE(LCTNRE),NUMPTS
      WRITE (6,40) (I,C(I),I=1,N)
      ERR=0.0
      DO 10 I=LCTNLE,LCTNRE
         TEMP=ABS(FTABLE(I)-HORNER(C,XTABLE(I),N))
         IF (TEMP.GT.ERR) ERR=TEMP
   10 CONTINUE
      WRITE (6,50) ERR
      DO 20 I=1,N
   20 CSTORE(I,NINT)=C(I)
      CSTORE(NPLUS1,NINT)=XTABLE(LCTNLE)
      RETURN
C
   30 FORMAT (//,5X, 15HINTERVAL NUMBER,I4, 16H WHICH BEGINS AT,E23.16,/
     1, 12H AND ENDS AT,E23.16,2X, 8HCONTAINS,I4, 8H POINTS.,/, 60H TH
     2E COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE,/)
   40 FORMAT ((10X, 2HC(,I2, 3H) =,E24.16))
   50 FORMAT (/, 47H THE ERROR OF APPROXIMATION IN THIS INTERVAL IS,E24.
     116, 1H.)
C
      END


      SUBROUTINE SETP(C,X,SMTH)
C
C  THIS SUBROUTINE APPROPRIATELY STORES IN THE ARRAY PPRIME THE VALUES
C  WHICH MUST BE INTERPOLATED TO GIVE THE PIECEWISE POLYNOMIAL THE
C  DESIRED SMOOTHNESS.
C
      DIMENSION C(18)
      COMMON /COMP/ CSTORE(18),B(150)
      COMMON/SCALAR/N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      COMMON /DDIF/ PPRIME(5),DUM1(18)
      INTEGER SMTH
C
      DO 10 I=1,N
   10 CSTORE(I)=C(I)
      NDUMMY=N
      I=0
   20 IF (I.GT.SMTH) RETURN
      IF (I.EQ.0) GO TO 30
      CALL DERIV(CSTORE,NDUMMY)
   30 PPRIME(I+1)=HORNER(CSTORE,X,NDUMMY)
      I=I+1
      GO TO 20
C
      END


      SUBROUTINE LSTINT(C,TOL,LNGTH,MAXNUM,ABORT)
C
C  THIS SUBROUTINE HANDLES THE SPECIAL CASE OF FINDING A SUBINTERVAL
C  AND A BEST APPROXIMATION ON THAT SUBINTERVAL WHEN THERE ARE TOO
C  FEW REMAINING POINTS FOR COMPUT TO WORK.
C
      INTEGER OLDLE,OLDRE
      LOGICAL TOOBIG,ABORT
      REAL C(18)
```

```
      COMMON X(1000),LCTNLE,LCTNRE,CSTORE(18,60)
      COMMON/SCALAR/N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      COMMON /COMP/ LCTNX(18)
C
      DO 10 OLDLE=1,NPLUS1
   10 CSTORE(OLDLE,NINT)=C(OLDLE)
      OLDLE=LCTNLE
      OLDRE=LCTNRE
      LCTNRE=MAXNUM
      LCTNLE=MIN0(MAXNUM-LNGTH+1,(MAXNUM-OLDLE+1)/2)-1
   20 LCTNLE=LCTNLE+1
      IF (MAXNUM-LCTNLE+1.LT.LNGTH) GO TO 40
      CALL SETP(CSTORE(1,NINT),X(LCTNLE),NLSMTH)
      NUMPTS=LCTNRE-LCTNLE+1
      CALL ASET(LCTNLE,LCTNRE,NLSMTH,N,C)
      CALL HOUSE(C(NLSMTH+2),TOL)
      CALL TOLCHK(TOL,TOOBIG)
      IF(TOOBIG)GO TO 20
      CALL FIX(C,X(LCTNLE),N)
      CALL STORE(CSTORE(1,NINT),OLDLE,LCTNLE)
      NINT=NINT+1
      DO 30 OLDLE=1,NPLUS1
   30 CSTORE(OLDLE,NINT)=C(OLDLE)
      RETURN
   40 ABORT=.TRUE.
      RETURN
C
      END


      SUBROUTINE COMPUT(C,TOL,LNGTH,MAXNUM.LAST,DONE,ABORT)
C
C  THIS SUBROUTINE FINDS THE LARGEST SUBINTERVAL AND THE BEST APPROX-
C  IMATION TO F ON THIS SUBINTERVAL SUCH THAT THE APPROXIMATION MEETS
C  THE DESIRED ERROR TOLERANCE ON THE SUBINTERVAL.
C
      INTEGER A,B
      LOGICAL LAST,OK,DONE,ABORT,TOOBIG
      REAL C(18)
      COMMON XTABLE(1000),LCTNLE,LCTNRE,CSTORE(18,60),CDERIV(300)
      COMMON/SCALAR/N,NPLUS1,NX,NXM1,NLSMTH,NRSMTH,NUMPTS,NINT
      COMMON/COMP/LCTNX(18),BB(150)
C
C  WE ASSUME THAT WE ARE CLOSE ENOUGH TO THE TRUE LARGEST SUBINTERVAL
C  RIGHT ENDPOINT WHEN WE KNOW THAT OUR APPROXIMATION TO THE TRUE RIGHT
C  ENDPOINT IS WITHIN ETA OF THE TRUE ENDPOINT.
C
      DATA ETA/.08/
C
      LITTLE=LCTNLE+LNGTH-1
      A=0
      LAST=.FALSE.
      CALL ASET(LCTNLE,LCTNRE,NLSMTH,N,C)
   10 NUMPTS=LCTNRE-LCTNLE
C
C  IF THE ACCURACY OF THE BEST APPROXIMATION ON THE CURRENT SUBINTERVAL
C  EXCEEDS TOL, CONTROL IS PASSED TO LINE 30.
C
      CALL HOUSE(C(NLSMTH+2),TOL)
      CALL TOLCHK(TOL,TOOBIG)
      IF(TOOBIG)GO TO 30
      IF(LCTNRE.LT.MAXNUM)GO TO 20
```

```
      CALL FIX(C,XTABLE(LCTNLE),N)
      DONE=.TRUE.
      RETURN
C
C  A IS THE CURRENT LARGEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT THE
C  BEST APPROXIMATION ON THIS SUBINTERVAL SATISFIES ALL CONSTRAINTS.
C
   20 A=LCTNRE
      IF ((XTABLE(B)-XTABLE(A).LE.ETA).OR.(B-A.LE.1)) GO TO 50
      GO TO 40
C
C  B IS THE CURRENT SMALLEST LOCATION FOR A RIGHT ENDPOINT SUCH THAT
C  THE BEST APPROXIMATION ON THIS SUBINTERVAL FAILS TO SATISFY THE CON-
C  STRAINTS.
C
   30 B=LCTNRE
   40 NEWTRY=(A+B)/2+1
      IF (NEWTRY.EQ.B) NEWTRY=NEWTRY-1
      IF(NEWTRY.LT.LITTLE) NEWTRY=LITTLE
      IF (NEWTRY.EQ.LCTNRE) GO TO 50
      LCTNRE=NEWTRY
      GO TO 10
C
C  IF A IS STILL 0, THEN NO SUBINTERVAL WITH AT LEAST LNGTH POINTS
C  WILL WORK, SO THE ALGORITHM IS TERMINATED.
C
   50 IF(A.NE.0)GO TO 60
      ABORT=.TRUE.
      RETURN
C
C  SINCE NEWTRY IS ALWAYS STRICTLY LESS THAN THE CURRENT B, IF NEWTRY=
C  LCTNRE, AND A IS NOT STILL 0, NEWTRY=A, WHICH IS A POINT WHICH SAT-
C  ISFIES ALL REQUIREMENTS.  WE NOW BACK THE RIGHT ENDPOINT OFF TO THE
C  BEST INTERIOR EXTREME POINT OF F-P TO ADD TO THE STABILITY OF THE
C  ALGORITHM.
C
   60 CALL FIX(C,XTABLE(LCTNLE),N)
      DO 70 I=1,N
         CDERIV(I)=C(I)
         CSTORE(I,NINT)=C(I)
   70 CONTINUE
      NDUMMY=N
      CALL DERIV(CDERIV,NDUMMY)
      I=NX
      LCTNRE=LCTNX(I)
      NEWTRY=LCTNRE
      SMLL=CMPR(CDERIV,NDUMMY,NEWTRY+LCTNLE-1,OK)
      IF(OK)GO TO 100
   80 I=I-1
      IF(I.EQ.0)GO TO 100
      NEWTRY=LCTNX(I)
      IF (NEWTRY.LT.LNGTH) GO TO 100
      TEMP=CMPR(CDERIV,NDUMMY,NEWTRY+LCTNLE-1,OK)
      IF(.NOT.OK)GO TO 90
      LCTNRE=NEWTRY
      GO TO 100
   90 IF(TEMP.GE.SMLL)GO TO 80
      SMLL=TEMP
      LCTNRE=NEWTRY
      GO TO 80
  100 LCTNRE=LCTNLE+LCTNRE-1
      IF (MAXNUM-LCTNRE+1.LT.LNGTH) LAST=.TRUE.
```

```
      RETURN
C
      END

      SUBROUTINE ASET(LCTNLE,LCTNRE,NLSMTH,N,C)
C
C  THIS SUBROUTINE SETS UP THE LINEAR SYSTEM WHICH IS SOLVED IN THE
C  L2 SENSE IN ORDER TO OBTAIN THE COEFFICIENTS OF THE POLYNOMIAL
C  APPROXIMATIONS.
C
      COMMON /HHOLDR/ A(150,10),BDATA(150,3)
      COMMON /DDIF/ PPRIME(5),DUM1(18)
      COMMON XTABLE(500),FTABLE(500)
      DIMENSION C(18)
C
      FAC=1.0
      I=1
      NM1=N-1
      NSMP1=NLSMTH+1
   10 IF (I.GT.NSMP1) GO TO 20
      C(I)=PPRIME(I)/FAC
      FAC=FLOAT(I)*FAC
      I=I+1
      GO TO 10
   20 XSTAR=XTABLE(LCTNLE)
      K=LCTNLE-1
      IF(NLSMTH.GE.0)K=K+1
      I=0
   30 K=K+1
      IF(K.GT.LCTNRE)GO TO 50
      I=I+1
      XNEW=XTABLE(K)-XSTAR
      BDATA(I,1)=XNEW
      IF(NLSMTH.LT.0)BDATA(I,2)=FTABLE(K)
      IF(NLSMTH.GE.0)BDATA(I,2)=FTABLE(K)-HORNER(C,XNEW,NSMP1)
      XPOWER=XNEW**NSMP1
      J=1
      JJ=NSMP1
   40 IF(JJ.GT.NM1)GO TO 30
      A(I,J)=XPOWER
      XPOWER=XNEW*XPOWER
      J=J+1
      JJ=JJ+1
      GO TO 40
   50 IF(NLSMTH.LT.0)A(1,1)=1.0
      RETURN
C
      END

      SUBROUTINE HOUSE(C,TOL)
C
C  THIS SUBROUTINE SOLVES A GIVEN LINEAR SYSTEM IN THE L2 SENSE USING
C  HOUSEHOLDER TRANSFORMATIONS.  THOUGH SOMEWHAT LENGTHY, IT WAS THE
C  BEST L2 ROUTINE AVAILABLE WHEN THIS PROGRAM WAS DEVELOPED.  AN ADAP-
C  TIVE CURVE FITTING PROGRAM WITH A MORE EFFICIENT L2 PACKAGE, AS WELL
C  AS AN L1 PACKAGE, IS CURRENTLY BEING DEVELOPED BY PAUL G. AVILA AND
C  G. T. TAYLOR.
C
      COMMON/COMP/LCTNX(18),B(150)
      COMMON /HHOLDR/ A,BDATA
```

```
      COMMON/SCALAR/DUM1(2),N,DUM2(3),M,DUM3                            HOUSE120
      REAL MAX                                                          HOUSE130
      DOUBLE PRECISION SUMM,X                                           HOUSE140
      DIMENSION X(10),KPIVOT(10),D(10),BETA(10),BETABR(10),SAVE(10),    HOUSE150
     $DP(10),DAA(10),A(150,10),AA(150,10),BB(150),BDATA(150,3),C(18)    HOUSE160
C                                                                       HOUSE170
      KRANK=N                                                           HOUSE180
      DO 10 J=1,N                                                       HOUSE190
         KPIVOT(J)=J                                                    HOUSE200
         DO 10 I=1,M                                                    HOUSE210
   10 AA(I,J)=A(I,J)                                                    HOUSE220
      DO 20 J=1,M                                                       HOUSE230
      B(J)=BDATA(J,2)                                                   HOUSE240
   20 BB(J)=BDATA(J,2)                                                  HOUSE250
      DO 130 K=1,N                                                      HOUSE260
         D(K)=0.0                                                       HOUSE270
         KCHNGE=K                                                       HOUSE280
         DO 40 JJ=K,N                                                   HOUSE290
            SUM=0.0                                                     HOUSE300
            DO 30 IA=K,M                                                HOUSE310
   30       SUM=SUM+AA(IA,JJ)*AA(IA,JJ)                                 HOUSE320
            IF(D(K).GE.SUM)GO TO 40                                     HOUSE330
            KCHNGE=JJ                                                   HOUSE340
            D(K)=SUM                                                    HOUSE350
   40    CONTINUE                                                       HOUSE360
C                                                                       HOUSE370
C     KCHNGE CONTAINS THE INDEX OF THE COLUMN OF GREATEST               HOUSE380
C     LENGTH BETWEEN K AND N.                                           HOUSE390
C                                                                       HOUSE400
         IF(KCHNGE.EQ.K)GO TO 60                                        HOUSE410
C                                                                       HOUSE420
C     START COLUMN INTERCHANGE.                                         HOUSE430
C                                                                       HOUSE440
         DO 50 I=1,M                                                    HOUSE450
            STORE=AA(I,KCHNGE)                                          HOUSE460
            AA(I,KCHNGE)=AA(I,K)                                        HOUSE470
            AA(I,K)=STORE                                               HOUSE480
   50    CONTINUE                                                       HOUSE490
         KK=KPIVOT(K)                                                   HOUSE500
         KPIVOT(K)=KPIVOT(KCHNGE)                                       HOUSE510
         KPIVOT(KCHNGE)=KK                                              HOUSE520
   60    CONTINUE                                                       HOUSE530
         IF(K.EQ.1)GO TO 70                                             HOUSE540
         MAX=ABS(D(1))                                                  HOUSE550
         TEST=(FLOAT(M-K+1)*1.0E-26)*(MAX*MAX)                          HOUSE560
         IF(ABS(D(K)).GT.TEST)GO TO 70                                  HOUSE570
         D(K)=SQRT(D(K))                                                HOUSE580
         KRANK=K-1                                                      HOUSE590
         GO TO 140                                                      HOUSE600
   70    CONTINUE                                                       HOUSE610
         AAKK=AA(K,K)                                                   HOUSE620
         SQDK=SQRT(D(K))                                                HOUSE630
         IF(AAKK.LT.0.0)GO TO 80                                        HOUSE640
         BETA(K)=1.0/(D(K)+AAKK*SQDK)                                   HOUSE650
         AA(K,K)=SQDK+AAKK                                              HOUSE660
         D(K)=-SQDK                                                     HOUSE670
         GO TO 90                                                       HOUSE680
   80    CONTINUE                                                       HOUSE690
         BETA(K)=1.0/(D(K)-AAKK*SQDK)                                   HOUSE700
         AA(K,K)=-SQDK+AAKK                                             HOUSE710
         D(K)=SQDK                                                      HOUSE720
   90    CONTINUE                                                       HOUSE730
```

```
      KT=K+1
      IF(K.EQ.N)GO TO 120
      DO 110 J=KT,N
         SAVE(J)=0.0
         DO 100 IA=K,M
100      SAVE(J)=SAVE(J)+AA(IA,K)*AA(IA,J)
         DO 110 I=K,M
            AA(I,J)=AA(I,J)-AA(I,K)*SAVE(J)*BETA(K)
110   CONTINUE
120   CONTINUE
130 CONTINUE
140 CONTINUE
    DO 150 I=1,KRANK
      II=I+1
      IF(I.EQ.N)GO TO 160
      DO 150 J=II,N
         AA(I,J)=AA(I,J)/D(I)
150 CONTINUE
160 CONTINUE
C
C  NOW ALL THE DIAGONAL ELEMENTS OF AA ARE 1 AND ALL OFF
C  DIAGONAL ELEMENTS OF AA ARE LESS THAN OR EQUAL TO 1.
C  RECALL AA(S,S)=1.0 WHICH IS STORED NOW IN DAA(S).
C
    DO 170 IS=1,KRANK
170 DAA(IS)=1.0
    IF(KRANK.EQ.N)GO TO 240
    DO 230 ISS=1,KRANK
      IS=KRANK-ISS+1
      KKR=KRANK+1
      SUM=0.0
      DO 180 IA=KKR,N
180   SUM=SUM+AA(IS,IA)*AA(IS,IA)
      SUM=SUM+1.0
      SQSUM=SQRT(SUM)
      BETABR(IS)=1.0/(SUM+SQSUM)
      DP(IS)=SQSUM+1.0
      DAA(IS)=-SQSUM
C
C  NOW W IS STORED IN ROW S FROM N-R+1 TO N WITH DIAGONAL
C  ELEMENTS STORED IN DP(S).
C  CALCULATE AA*W.
C
      KKS=IS-1
      KSS=KRANK+1
      IF(IS.EQ.1)GO TO 220
      DO 210 J=1,KKS
         SAVE(J)=0.0
         DO 190 IA=KSS,N
190      SAVE(J)=SAVE(J)+AA(IS,IA)*AA(J,IA)
         SAVE(J)=SAVE(J)+DP(IS)*AA(J,IS)
         AA(J,IS)=AA(J,IS)-DP(IS)*SAVE(J)*BETABR(IS)
         DO 200 I=KSS,N
C
C  NOTE...W(I) IS STORED AT AA(IS,I).
C
200      AA(J,I)=AA(J,I)-AA(IS,I)*SAVE(J)*BETABR(IS)
210   CONTINUE
220   CONTINUE
230 CONTINUE
240 CONTINUE
    ICOUNT=1
```

```
      DO 250 I=1,N
  250 X(I)=0.0
  260 CONTINUE
C
C  PREMULTIPLY BY THE HOUSEHOLDER TRANSFORMATIONS.
C
      DO 290 I=1,KRANK
         SUM=0.0
         DO 270 IA=I,M
  270    SUM=SUM+AA(IA,I)*B(IA)
         SUM=SUM*BETA(I)
         DO 280 J=I,M
            B(J)=B(J)-AA(J,I)*SUM
  280    CONTINUE
  290 CONTINUE
C
C  NOW ONLY USE THE FIRST KRANK TERMS OF B.
C  CALCULATE (D INVERSE)*B .
C
      DO 300 I=1,KRANK
         B(I)=B(I)/D(I)
  300 CONTINUE
C
C  NOW SOLVE (V INVERSE)*B .
C
      DO 320 II=1,KRANK
         I=KRANK+1-II
         B(I)=B(I)/UAA(I)
         KK=I-1
         IF(I.EQ.1)GO TO 320
         DO 310 J=1,KK
            B(J)=B(J)-AA(J,I)*B(I)
  310    CONTINUE
  320 CONTINUE
      DO 340 I=1,N
         IF(I.LE.KRANK)GO TO 330
         B(I)=0.0
  330    CONTINUE
  340 CONTINUE
      IF(KRANK.EQ.N)GO TO 380
C
C  MULTIPLY BY P FOR I=KRANK TO 1.
C
      KK=KRANK+1
      DO 370 I=1,KRANK
         SUM=0.0
         DO 350 IA=KK,N
  350    SUM=SUM+AA(I,IA)*B(IA)
         SUM=(SUM+B(I)*DP(I))*BETABR(I)
         B(I)=B(I)-DP(I)*SUM
         DO 360 J=KK,N
            B(J)=B(J)-AA(I,J)*SUM
  360    CONTINUE
  370 CONTINUE
  380 CONTINUE
C
C  TEST FOR CONVERGENCE.
C  FIRST TEST, TOO MANY ITERATIONS.
C  SECOND TEST, SEE IF X IS DECREASING.
C
      SUM=0.0
      DO 390 I=1,N
```

```
  390 SUM=SUM+B(I)*B(I)
      IF(ICOUNT.EQ.1)GO TO 400
      IF(SUM.LE..5*TEST)GO TO 410
      ICOUNT=11
      GO TO 410
  400 TESTI=SUM
  410 TEST=SUM
      DO 420 I=1,N
         KP=KPIVOT(I)
         X(KP)=B(I)+X(KP)
  420 CONTINUE
C
C  CALCULATE A*X-B .
C
      DO 440 I=1,M
         SUMM=0.0
         DO 430 J=1,N
  430    SUMM=SUMM+A(I,J)*X(J)
  440 B(I)=3B(I)-SUMM
C
C  THIRD TEST, WAS THE CORRECTION SIGNIFICANT.
C
      IF(TEST.LT.1.E-30*TESTI)GO TO 470
      IF(ICOUNT.EQ.5)GO TO 450
      IF(ICOUNT.GE.6)GO TO 470
      ICOUNT=ICOUNT+1
      GO TO 260
  450 CONTINUE
      WRITE(6,460)
  460 FORMAT(1X,35HCOMPLETED 10 ITERATIONS AND STOPPED)
  470 CONTINUE
      DO 480 J=1,N
  480 C(J)=X(J)
      RETURN
C
      END


      SUBROUTINE TOLCHK(TOL,TOOBIG)
C
C  THIS SUBROUTINE CHECKS TO SEE IF THE CURRENT APPROXIMATION MEETS
C  THE SPECIFIED TOLERANCE.  IF NOT, WE RETURN TO COMPUT TO REDUCE THE
C  INTERVAL LENGTH.  IF TOL IS MET, THE EXTREME POINTS OF THE APPROX-
C  IMATION APE STORED IN THE ARRAY LCTNX.
C
      COMMON/COMP/LCTNX(18),B(150)
      COMMON/SCALAR/DUM1(2),N,DUM2(3),M,DUM3
      REAL MAX
      LOGICAL TOOBIG
C
      TOOBIG=.TRUE.
      MAX=0.0
      DO 10 I=1,M
      TEMP=ABS(B(I))
      IF (TEMP.GT.MAX) MAX=TEMP
   10 CONTINUE
      IF(MAX.GT.TOL)RETURN
      TOOBIG=.FALSE.
      BIG=ABS(B(M))
      K=M
      LCTN=M
      SGN=SIGN(1.0,B(M))
```

```
      DO 60 J=1,N
         I=N+1-J
   20    IF(K.LT.1)GO TO 30
         TEMP=SGN*B(K)
         IF(TEMP.GT.0.0)GO TO 40
         SGN=SIGN(1.0,B(K))
   30    LCINX(I)=LCTN
         BIG=-TEMP
         LCTN=K
         K=K-1
         GO TO 60
   40 IF(TEMP.LT.BIG)GO TO 50
         LCTN=K
         BIG=TEMP
   50    K=K-1
         GO TO 20
   60 CONTINUE
      RETURN
C
      END


      SUBROUTINE FIX(C,XSTAR,N)
C
C  THIS SUBROUTINE TAKES A POLYNOMIAL EXPRESSED IN NEWTONS FORM AND
C  FORMS THE COEFFICIENTS OF THAT SAME POLYNOMIAL IN STANDARD FORM.
C
      DIMENSION C(18)
C
      NM1=N-1
      DO 20 J=1,NM1
         K=N-J
         DO 10 I=K,NM1
            C(I)=C(I)-XSTAR*C(I+1)
   10    CONTINUE
   20 CONTINUE
      RETURN
C
      END


      SUBROUTINE DERIV(C,N)
C
C  THIS SUBROUTINE REPLACES THE COEFFICIENTS OF A POLYNOMIAL IN STAND-
C  ARD FORM WITH THE COEFFICIENTS OF THE DERIVATIVE OF THIS POLYNOMIAL.
C  THE NUMBER OF COEFFICIENTS, N, IS DECREMENTED.
C
      DIMENSION C(N)
C
      N=N-1
      DO 10 I=1,N
   10 C(I)=FLOAT(I)*C(I+1)
      RETURN
      END


      FUNCTION CMPR(C,N,NEWTRY,OK)
C
C  THIS SUBROUTINE COMPARES THE FIRST DERIVATIVE OF THE CURRENT PIECE OF
C  THE PIECEWISE POLYNOMIAL APPROXIMATION EVALUATED AT XTABLE(NEWTRY)
C  WITH THE FIRST DERIVATIVE OF THE QUADRATIC INTERPOLATION OF F, CEN-
C  TERED AROUND XTABLE(NEWTRY), EVALUATED AT XTABLE(NEWTRY).  IF THESE
```

```
C  TWO DIFFER IN ABSOLUTE VALUE BY LESS THAN TOLER (EITHER ABSOLUTELY
C  OR RELATIVELY), WE SET OK TO .TRUE. AND WE ACCEPT XTABLE(NEWTRY) AS
C  A REASONABLE SUBINTERVAL RIGHT ENDPOINT.
C
      LOGICAL OK
      COMMON X(500),F(500),DUM1(1382)
      DIMENSION C(18)
      DATA TOLER/.05/
C
      OK=.FALSE.
      A=(F(NEWTRY)-F(NEWTRY-1))/(X(NEWTRY)-X(NEWTRY-1))
      B=(F(NEWTRY+1)-F(NEWTRY))/(X(NEWTRY+1)-X(NEWTRY))
      D=(B-A)/(X(NEWTRY+1)-X(NEWTRY-1))
      CMPR=A+D*(X(NEWTRY)-X(NEWTRY-1))
      A=TOLER*CMPR
      CMPR=ABS(CMPR-HORNER(C,X(NEWTRY),N))
      IF (CMPR.LE.A) OK=.TRUE.
      RETURN
C
      END


      FUNCTION EVAL(X)
C
C  THIS FUNCTION EVALUATES THE PIECEWISE POLYNOMIAL APPROXIMATION
C  AT ANY POINT IN THE ENTIRE INTERVAL.
C
      COMMON DUMMY(1002),CSTORE(18,60),DUM(300)
      COMMON/SCALAR/N,NPLUS1,DUM2(5),NINT
C
      IF(NINT.LT.2)GO TO 20
      DO 10 I=2,NINT
         ISTORE=I-1
         IF(X.LE.CSTORE(NPLUS1,I))GO TO 30
   10 CONTINUE
      ISTORE=NINT
      GO TO 30
   20 ISTORE=1
   30 EVAL=HORNER(CSTORE(1,ISTORE),X,N)
      RETURN
C
      END


      FUNCTION HORNER(C,X,N)
C
C  THIS FUNCTION EVALUATES A POLYNOMIAL IN STANDARD FORM BY HORNERS
C  METHOD.
C
      DIMENSION C(N)
C
      HORNER=C(N)
      I=N
   10 IF (I.LT.2) RETURN
      HORNER=HORNER*X+C(I-1)
      I=I-1
      GO TO 10
C
      END
```

LEAST SQUARES ADAPTIVE CURVE FITTING PROGRAM : SAMPLE RUN

INPUT :

```
        6     2    3.00         ( N, SMTH, TOL )

        3.0        0.0          (XTABLE, FTABLE)
        5.0        1.3
        7.0        3.4
       11.0        5.2
       13.0        6.0
       15.0       14.4
       17.5       21.4
       20.0       27.4
       22.5       50.9
       25.0       49.3
       27.5       47.5
       30.0       51.5
       35.0       36.5
       40.0       27.9
       50.0        9.4
       60.0        4.2
```

OUTPUT :

```
     INTERVAL NUMBER  1 WHICH BEGINS AT  .3000000000000000E+01
AND ENDS AT  .2185714285714278E+02  CONTAINS 133 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

          C( 1) = -.3189942743572442E+02
          C( 2) =  .1812103641220483E+02
          C( 3) = -.3560852823483785E+01
          C( 4) =  .3242284790553871E+00
          C( 5) = -.1350234902068514E-01
          C( 6) =  .2158257913951744E-03

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS  .287469301288797E+01.


     INTERVAL NUMBER  2 WHICH BEGINS AT  .2185714285714278E+01
AND ENDS AT  .2528571428571411E+02  CONTAINS  25 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

          C( 1) =  .7525331567081399E+06
          C( 2) = -.1568254831136037E+06
          C( 3) =  .1303544628337352E+05
          C( 4) = -.5402174434562294E+03
          C( 5) =  .1116296306156516E+02
          C( 6) = -.9202040460845584E-01

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS  .2824472463024449E+01.


     INTERVAL NUMBER  3 WHICH BEGINS AT  .2528571428571411E+02
AND ENDS AT  .3557142857142844E+02  CONTAINS  73 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE
```

```
C( 1) =  .6867241093987040E+05
C( 2) = -.1121136988075014E+05
C( 3) =  .7277415199334O6E+03
C( 4) = -.2346667791438097E+02
C( 5) =  .3759555635732301E+00
C( 6) = -.2395099334129056E-02
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .2985086750638296E+01.

INTERVAL NUMBER 4 WHICH BEGINS AT .3557142857142844E+02
AND ENDS AT .5399999999999977E+02 CONTAINS 130 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) = -.8446628095918451E+04
C( 2) =  .9702467401092472E+03
C( 3) = -.4389461668285185E+02
C( 4) =  .9842698198241457E+00
C( 5) = -.1096799498228268E-01
C( 6) =  .4863340401267810E-04
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .1464785127359050E+01.

INTERVAL NUMBER 5 WHICH BEGINS AT .5399999999999977E+02
AND ENDS AT .5999999999999977E+02 CONTAINS 43 POINTS.
THE COEFFICIENTS OF BEST APPROXIMATION IN THIS INTERVAL ARE

```
C( 1) = -.1771038129739963E+07
C( 2) =  .1591827758229449E+06
C( 3) = -.5718444554639893E+04
C( 4) =  .1026330177930108E+03
C( 5) = -.9202908099941851E+00
C( 6) =  .3298242672003487E-02
```

THE ERROR OF APPROXIMATION IN THIS INTERVAL IS .9836415372650436E+00.